Franzis
Computer-Praxis

Klein

Rechner modular

Der NDR-Klein-Computer –
selbstgebaut und programmiert



Franzis'

Klein
Rechner modular

In der Reihe
Franzis Computer-Praxis
sind erschienen:

Benda, Mikrocomputer-Technik praxisnah
Busch, Basic für Aufsteiger
Busch, Basic für Einsteiger
Busch, Der sichere Einstieg in Pascal
Esders, Assembler-Programme zum Apple II
Esders, Das Buch zum Apple II
Feichtinger, Mit Computern steuern
Haugg, Software-Engineering
Janson, Die beiden Datenbanksysteme dBase II und III
Klein, Z-80 Applikationsbuch
Klein, Mikrocomputer selbstgebaut und programmiert
Klein, Mit HEXMON Programme entwickeln
Klein, Die Prozessoren 68000 und 68008
Klein, Was ist Pascal
Link, Messen, Steuern und Regeln mit Basic
Merker, Hardware-Erweiterung für den Apple II
Merker, Hardware-Erweiterung für den ZX 81
Miedel/Kotulla, Das große CPC-Arbeitsbuch
Piotrowski, IEC-Bus
Plate, Anwenderhandbuch CP/M-68 K
Plate, Betriebssystem CP/M
Plate, Computergrafik: Einführung – Algorithmen – Programmentwicklung
Plate/Lecher, Hard- und Software für den Epson HX-20
Plate/Wittstock, Pascal: Einführung – Programmentwicklung – Strukturen
Pütz, Praxis der Datenübertragung
Pütz, Das große C-64-Handbuch
Röckrath, Microsoft-Basic: Konzepte, Algorithmen, Datenstrukturen
Ruhland, DOS 3.3 - das Diskettenbetriebssystem des Apple II
Schirrmacher, MacIntosh programmieren
Schoffa, Die Programmiersprache LISP
Stenzel, Maschinensprache? - kein Problem!
Troitzsch, Mikrocomputer-Schaltungstechnik
Wunderlich, Erfolgreicher mit CBM arbeiten
Wunderlich, Erfolgreicher mit dem VC 64 arbeiten
Zech, Die Programmiersprache Forth
Zilker, Praxis des Multitasking

Franzis Computer-Praxis

Rolf-Dieter Klein

Rechner modular

Der NDR-Klein-Computer – selbstgebaut
und programmiert

Mit 410 Abbildungen und 25 Tabellen



CIP-Kurztitelaufnahme der Deutschen Bibliothek

Klein, Rolf-Dieter:
Rechner modular: Der NDR-Klein-Computer-selbstgebaut und programmiert / Rolf-Dieter Klein. – München: 1987.
(Franzis-Computer-Praxis)
ISBN 3-7723-8721-7

Satz und Druck: Kösel, Kempten

Printed in Germany · Imprimé en Allemagne.

ISB N 3-7723-8721-7

Vorwort

Die Mikrocomputer dringen in alle Bereiche des Lebens ein. Daher kann es nur von Nutzen sein, sich mit dieser Technik vertraut zu machen. Für den Anfänger ist es aber schwierig, einen Einstieg zu finden. Es gibt eine Vielzahl fertiger Computer, die aber meist nur wenig durchschaubar sind. Schaltpläne sind nur selten für den Anwender erhältlich. Außerdem verwenden die Hersteller oft eigene integrierte Bausteine, deren Innenleben ein streng gehütetes Geheimnis bleibt. Das gleiche gilt für die Software.

In dem vorliegenden Buch wird ein Mikrocomputersystem vorgestellt, das dieses Übel beseitigen soll. Alle Mikrorechnerschaltungen sind durch im Handel erhältliche Leiterplatten unterstützt. Sogar eine Fernsehreihe existiert, die dieses System verwendet. Der hier vorgestellte Mikrorechner verdankt seinen Namen dieser Serie und heißt NDR-Klein-Computer.

Der Computer ist modular aufgebaut, das heißt, durch Kombination von verschiedenen Baugruppen kann man den Computer ganz nach Bedarf ausbauen. Das geht vom einfachen Z80-Rechner mit 4 KByte über einen voll ausgebauten Z80-Computer mit 1 MByte sogar bis zu einem 68020-Computer mit 4 MByte und Winchester. In diesem Buch wird der Aufbau des Z80-Computers behandelt, der am Schluß mit Floppy-Laufwerken ausgestattet werden kann und damit keinen Vergleich mit anderen kommerziellen Systemen zu scheuen braucht. Durch die Verwendung des CP/M-Betriebsystems ist auch eine Vielfalt an kompatibler und preisgünstiger Software verfügbar.

Besonderer Dank gilt auch Herrn Dr. Hans Hehl für die Durchsicht des Manuskripts. Durch das Sammeln seiner Erfahrungen beim Einsatz des NDR-Klein-Computers im Gymnasium Markt-Schwaben hat er wertvolle Beiträge zu diesem Buch geleistet.

Rolf-Dieter Klein, München

Wichtiger Hinweis

Inhalt

1 Spannungsversorgungen

Ohne Energie geht nichts, so auch bei unserem Mikrocomputer, den wir bauen wollen. Man sollte die Bedeutung der Spannungsversorgung nicht unterschätzen, sie schafft manchmal ungeahnte Probleme.

Mikrocomputer benötigen im allgemeinen zunächst einmal eine Versorgungsspannung von 5 V. Für manche Zusatzgeräte wird auch noch je eine Spannung von + 12 V und − 12 V gebraucht.

Die Stromaufnahme unseres Computers liegt zwischen 2 A und 5 A (bei 5 V) je nach Ausbaustufe.

In diesem Kapitel werden wir eine kleine Spannungsversorgung aufbauen, die für die ersten Versuche ausreicht und ca. 3 A liefern kann. Zum Aufbau der nachfolgenden Schaltung wird ein einfaches Vielfachmeßinstrument mit Drehspulmeßwerk benötigt.

1.1 5-V-Versorgung

Mikrorechner sind sehr wählerisch, was die Energieversorgung angeht. Sie wollen eine oder mehrere Gleichspannungen haben, die bestimmte Werte genau einhalten müssen. So benötigt der NDR-Klein-Mikrorechner einer 5-V-Spannung, die sehr enge Toleranzen einhalten muß. Die Spannung darf nicht größer als 5,25 V sein, aber auch nicht kleiner als 4,75 V.

Wenn die Spannung nämlich zu groß wird, können Bauteile beschädigt werden. Ist sie zu niedrig, so arbeitet der Rechner nicht korrekt und liefert fehlerhafte Ergebnisse. Man kann also weder die Netzspannung von 220 V direkt für den Computer verwenden, noch kann man eine Taschenlampenbatterie als Energiequelle nutzen, da diese die geforderte Spannungstoleranz im Betrieb nicht einhält. Die Baugruppe POW5V (zusammen mit einem Netztransformator) löst das Energieversorgungsproblem.

Zunächst muß aus der lebensgefährlichen 220-V-Netzspannung eine harmlose Niederspannung von etwa 7,5 V bis 12 V gemacht werden. Es gibt eine Reihe von Transformatoren im Handel, die man dazu verwenden kann. Hier die Daten für den benötigten Trafo:

Eingangsspannung: 220 V
Ausgangsspannung: 7,5 V bis max. 12 V
(am besten in Stufen einstellbar)
Leistung: ca. 30 Watt
oder Strom: ca. 3 Ampere
VDE-Zeichen.

Man kann ihn im Fachhandel oder bei den Baugruppen-Lieferanten besorgen. Am besten wäre eine Ausführungsform mit geschlossenem Gehäuse, bei der man die lebensgefährliche 220-V-Spannung nicht berühren kann.

Erstes Experiment

Folgendes Experiment ist dann ganz gefahrlos:

1. Der Trafo wird ans Netz angeschlossen (Achtung: VDE-Vorschriften beachten).
2. Das Meßgerät wird auf einen Wechselspannungs-Meßbereich gestellt, dessen Maximalausschlag über 12 V liegt.
3. Die Meßspitzen werden an die Trafo-Ausgangs-Klemmen angelegt. Jetzt muß man eine Spannung zwischen 7,5 V und 12 V ablesen können.
4. Wenn man ein Oszilloskop als Meßgerät verwendet, so erscheint auf dem Bildschirm eine sinusförmige Wechselspannung mit positivem und negativem Spannungsteil. Die Spannung zwischen der Nullinie und der Spitze der Wechselspannung ist höher als die auf dem Vielfachmeßgerät angezeigte Spannung *(Abb. 1.1.1)*

Man bezeichnet die Spannung zwischen Nullinie und Spitze der Sinuskurve als Spitzenspannung, während ein Vielfachmeßgerät die sogenannte effektive Spannung anzeigt. Die Spitzenspannung steht nämlich nur ganz kurz während einer jeden Periode zur Verfügung. Das träge Meßwerk eines Zeigerinstruments kann nicht bis dahin ausschlagen, sondern registriert nur den effektiven Wert der Spannung. Die effektive Spannung kann man aus der Spitzenspannung ausrechnen. Dazu muß man den Wert der Spitzenspannung durch $\sqrt{2}$ (ungefähr 1,4142) dividieren.

$$\text{Ueff} = \frac{1}{\sqrt{2}} \cdot \text{Uspitze.}$$

Abb. 1.1.1 zeigt den Verlauf einer Wechselspannung, wie man sie auf einem Oszilloskop sehen könnte. Jede Wechselspannung besitzt positive und negative Spannungsteile. Das ist für Gleichstrom-Geräte ungeeignet, da diese durch falsch gepolte Spannungen oft sogar beschädigt werden können. Ein Gleichrichter kann in solchen Fällen eingesetzt werden, um aus einer Wechselspannung eine Gleichspannung zu machen. Heute verwendet man dazu Brückengleichrichter, die aus vier Dioden bestehen.

Abb. 1.1.2 zeigt, wie das Signal hinter dem Gleichrichter aussehen soll.
Abb. 1.1.3 zeigt den Schaltplan der POW5V-Baugruppe,
Abb. 1.1.4 zeigt den Bestückungsplan und *Abb. 1.1.5* die fertige Baugruppe.
Tabelle 1.1.1 zeigt die Stückliste.
Abb. 1.1.6 zeigt die Leiterbahnseite der POW5V.

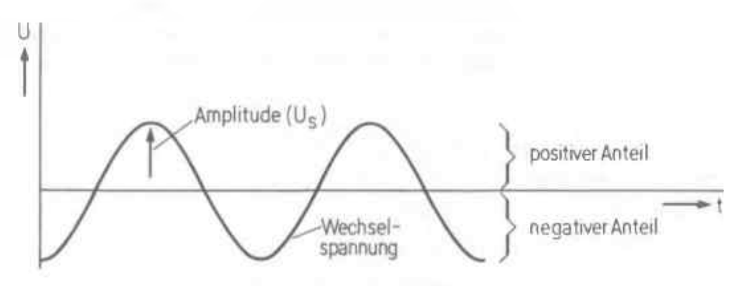


Abb. 1.1.1 Das Oszillogramm einer Wechselspannung

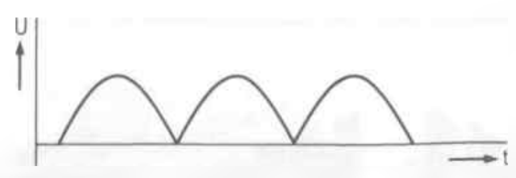


Abb. 1.1.2 Ein Brückengleichrichter "klappt" die negativen Halbwellen der Wechselspannung nach oben um

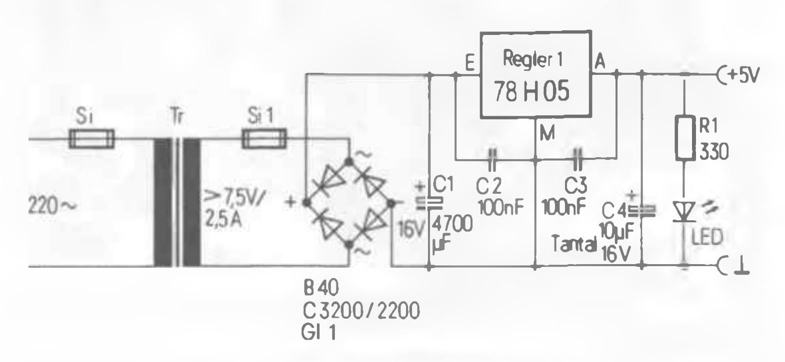


Abb. 1.1.3 Der Schaltplan der POW5V

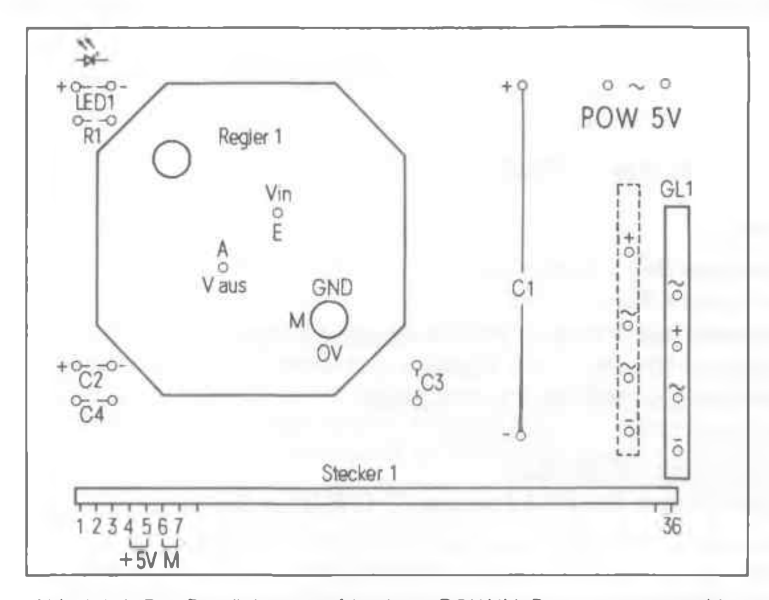


Abb. 1.1.4 Der Bestückungsaufdruck zu POW5V. Da es zwei verschiedene Bauarten des Gleichrichters gibt, mit verschiedenen Anordnungen der Anschlußfahnen, sind zwei Einbaulagen gekennzeichnet. Der Gleichrichter sitzt richtig, wenn seine Markierungen der Anschlüsse mit denen an den Platinenbohrungen für ihn übereinstimmen.

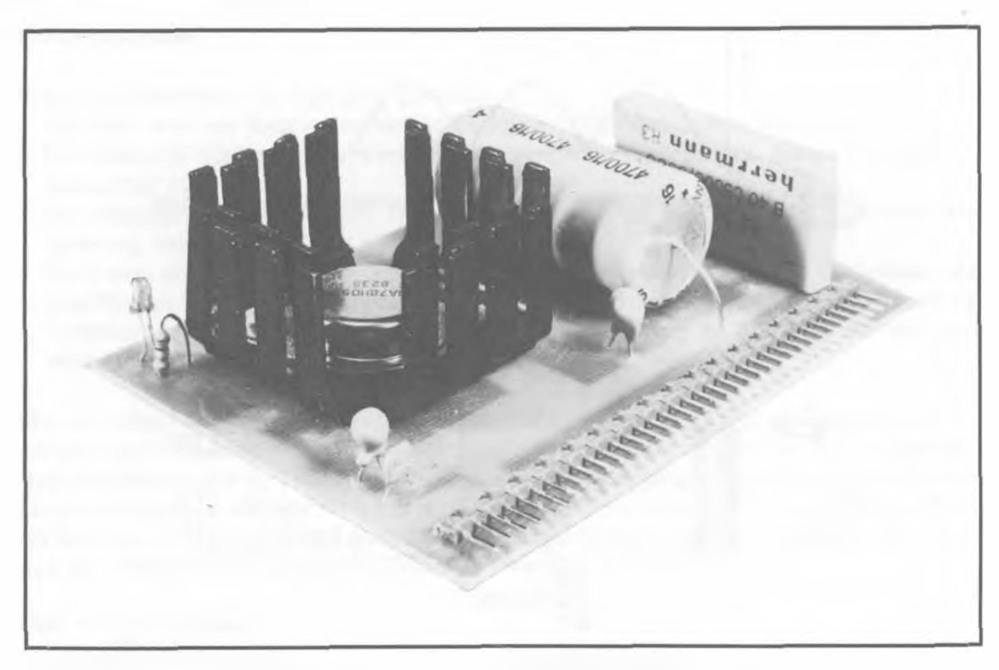


Abb. 1.1.5 Das fertige Werk

Tabelle 1.1.1 Stückliste zur POW5V

2A-Version

1 x Gleichrichter B40 C3200/2200
1 x Kondensator 4700 µF 16V
2 x Kondensator 100 nF ca. 100V keramische Scheibe
1 x Kondensator 10 µF 16V Tantal
1 x Spannungsregler 78H05 im TO-3 Gehäuse
1 x Widerstand 1/4 W 330 Ohm
1 x LED rot, 3 mm Durchmesser
1 x Fingerkühlkörper mit TO-3 Lochung 6° C/W Höhe 25.4 mm

5A-Version

wie oben, jedoch
1 x Gleichrichter B40 C5000/3300 mit Kühlung
1 x Kühlkörper mit TO-3 Lochung ca. 2° C/W
Montage des Reglers außerhalb der Platine

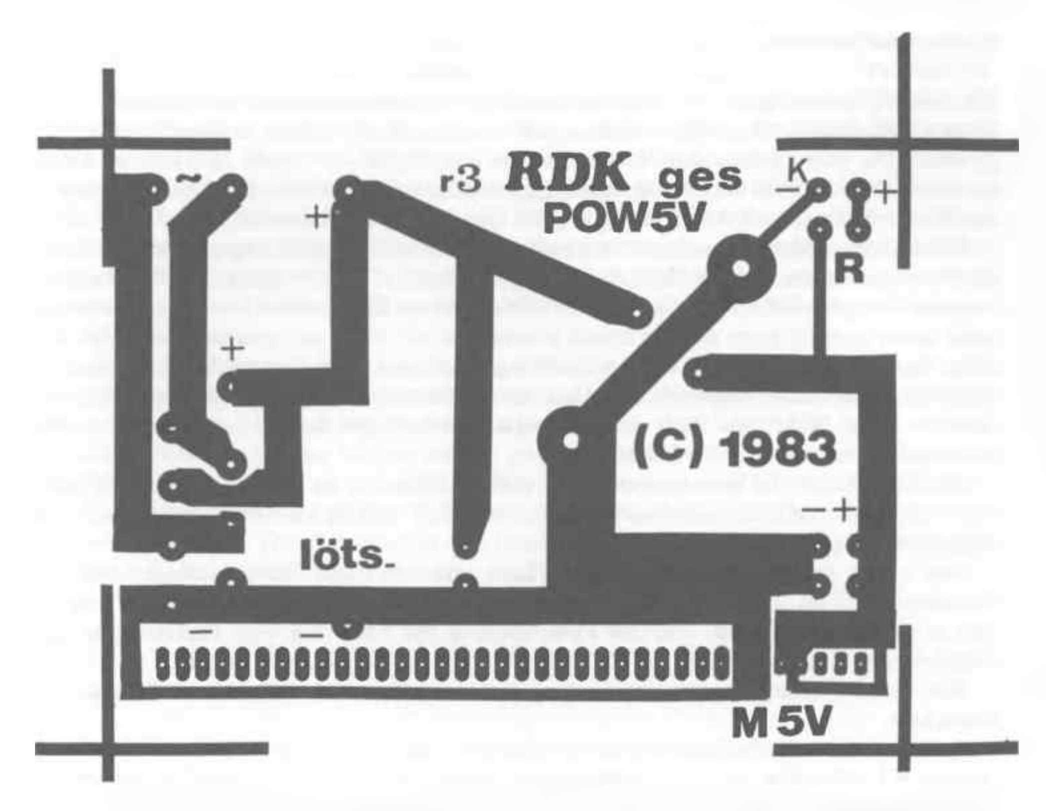


Abb. 1.1.6 Lötseite der Leiterplatte POW5V

Zum Aufbau

Der Gleichrichter wird als erstes auf die Leiterplatte POW5V gelötet. Dabei sollte man darauf achten, daß der Gleichrichter nicht verkehrt eingesteckt wird: Auf der Leiterplatte befindet sich eine Plus-Markierung, dort muß der mit „+“ gekennzeichnete Gleichrichteranschluß eingesteckt werden. Der Transformatorausgang wird mit zwei Leitungen mit dem Wechselspannungseingang der Leiterplatte verbunden. Im Schaltbild ist noch eine Sicherung mit Si1 eingezeichnet. Diese Sicherung ist im allgemeinen im Trafo enthalten, wenn er VDE-mäßig aufgebaut ist (also mit Gehäuse und Anschlußleitung).

Die Sicherung ist daher nicht auf der Baugruppe eingeplant. Der Wechselspannungseingang der POW5V-Baugruppe ist mit einem Wellensymbol „~“ gekennzeichnet.

Man kann die Leitungen von oben durch die Bohrungen der Leiterplatte stecken und unten auf der Lötseite verlöten. Man kann auch Lötstifte durch die Bohrungen stecken und die Zuleitungen an den Stiften festlöten. Wenn man 1,3-mm-Lötstifte verwenden will, muß man die 1-mm-Platinenbohrungen aufbohren. Es gibt aber auch 1-mm-Lötstifte im Handel, die man ohne Umstände einlöten kann.

Ein zweites Experiment

Mit dem Vielfachmeßgerät. Die Ausgangsspannung des Gleichrichters soll kontrolliert werden. Dazu wird das Meßgerät auf Gleichspannungsmessung eingestellt und ein Meßbereich über 20 V gewählt. Die Masseleitung des Meßgerätes, die mit „COM“, „–“ oder „0 Volt“ am Gerät gekennzeichnet ist, wird mit dem Minusausgang des Gleichrichters verbunden. Der Plus-Eingang des Meßgerätes wird mit dem Plusausgang des Gleichrichters verbunden.

Der angezeigte Meßwert muß jetzt in etwa dem aus dem ersten Versuch entsprechen. Sollte ein Ergebnis ausbleiben, so kann man einen Widerstand parallel zum Ausgang des Gleichrichters schalten. Man nehme dazu zum Beispiel den Widerstand von 330 Ω, der im Bausatz vorhanden ist (aber später noch für einen anderen Zweck gebraucht wird). Wenn sich jetzt kein befriedigender Ausschlag zeigt, dann sollte man alle Verbindungen nochmals genau überprüfen. Wenn man ein Oszilloskop verwendet, muß man unbedingt diesen Widerstand (330 Ω, ¼ oder ⅛ W) parallelschalten. Ohne Widerstand fließt nämlich praktisch kein Strom durch den Gleichrichter. Das Meßergebnis wird deshalb verfälscht.

Mit dem Oszilloskop kann man erkennen, daß die Spannung am Gleichrichterausgang noch sehr wellig ist. Sie sinkt zwischendurch kurz auf Null Volt ab und steigt fast bis auf den Spitzenwert der Wechselspannung an.

Eine solche pulsierende Gleichspannung kann man mit einem Elektrolytkondensator, im Fachjargon auch als „Elko“ bezeichnet, glätten. Beim Elko muß man beim Einbau darauf achten, daß er richtig gepolt wird, also der Plus-Anschluß des Elkos mit dem Plus-Ausgang des Gleichrichters verbunden wird. Auf der Leiterplatte ist das entsprechend markiert.

Wird ein Elko verkehrt herum eingebaut, so wird er zerstört. Im Schaltbild ist der Elko mit C1 bezeichnet.

Drittes Experiment

Mit einem Vielfachmeßinstrument: Die Ausgangsspannung hinter dem Elko liegt höher als bei der Messung ohne Elko (ca. um den Faktor 1.4), denn nun liegt eine fast glatte Spannung an. Der Elko wird bis zum Spitzenwert aufgeladen.

Messung mit dem Oszilloskop. Die Spannung am Elko ist praktisch eine glatte Linie. Auf dem Schirm sieht man jetzt normalerweise keine Welligkeit mehr. Wenn man aber eine Last (zum Beispiel eine Lampe) zum Elko parallelschaltet, so beginnt die vorher glatte Linie wieder Wellenform anzunehmen. Der nächste Schritt ist der Einbau eines Spannungsreglers. Dieser hat die Aufgabe, aus der Gleichspannung von über 7,5 V eine exakte 5-V-Spannung zu machen.

So ein Spannungsregler, wie wir ihn verwenden, enthält in seinem Inneren eine Vielzahl von Transistoren und Widerständen, es ist ein integrierter Schaltkreis. Er prüft durch einen Spannungsvergleich in seinem Inneren, ob die Ausgangsspannung dem Soll entspricht, also exakt 5 V hat oder nicht. Ist die Spannung am Ausgang geringfügig abgesunken, so hebt er sie sofort wieder an und umgekehrt. Dazu benötigt er aber am Eingang eine Spannung, die größer sein muß als die von ihm geregelte Ausgangsspannung. Meist reichen 7,5 V effektive Eingangsspannung dazu gerade noch aus. Ist die Eingangsspannung niedriger als dieser Wert, so kann der Regler nicht mehr regeln, weil nicht mehr genug Spannungsreserve vorhanden ist, und am Ausgang sinkt die Spannung ab. Die Eingangsspannung darf also auch kurzzeitig nicht unter diesem Wert liegen, denn dann sinkt die Ausgangsspannung ebenfalls unter 5 V. Am Eingang des Spannungsreglers sollten also stets mehr als 7,5 V anliegen.

Andererseits gibt es auch Schwierigkeiten, wenn zuviel Eingangsspannung am Regler anliegt. Der Spannungsregler muß die Spannungsdifferenz zwischen Eingang und Ausgang in seinem

Inneren vernichten. Beispiel: Am Eingang liegen 9 V, am Ausgang 5 V. Die Differenz beträgt somit 4 V. Bei einem Strom von 1 A werden 1 A · 4 V in Wärme umgesetzt (U*I = P), also 4 W.

Die Energie, die der Spannungsregler in Wärme umsetzt, ist um so größer, je höher die Eingangsspannung ist und je mehr der Spannungsregler belastet wird, also je mehr Baugruppen angeschlossen sind. Dann wird der Regler einfach sehr sehr warm.

Eine im Regler eingebaute Temperatursicherung schaltet den Regler rechtzeitig ab, ehe er zu heiß wird. Jedoch liefert er dann auch keine 5 V mehr, der angeschlossene Computer bleibt stehen. Es ist also gar nicht so einfach mit dem Regler. Jedoch nicht verzagen, Ausprobieren zeigt, daß alles gut funktioniert. Der Regler wird auf einem Kühlkörper montiert, der die Wärme besser an die Umwelt abführen soll. Auf der POW5V-Baugruppe werden Kühlkörper und Regler mit Schrauben befestigt.

Achtung: Die Anschlußbeinchen des Reglers dürfen den Kühlkörper nicht berühren, ggf. sollte man Isolierhülsen verwenden. Die Anschlußbeine sind am Regler asymmetrisch angebracht. Der Regler paßt also nur in einer Lage auf die Platine.

Zum sicheren Betrieb des Reglers werden noch drei kleinere Kondensatoren benötigt. Im Schaltbild sind sie mit C2, C3 und C4 bezeichnet. Die Kondensatoren C2 und C3 sind 100-nF-Kondensatoren, deren Polung keine Rolle spielt, und C4 ist ein kleiner Elektrolytkondensator oder ein sogenannter Tantalkondensator von 10 µF, der die Ausgangsspannung von Störungen befreien soll. Bei seinem Einbau ist wieder die Plusmarkierung zu beachten.

Ein viertes Experiment

Es wird am Vielfachmeßinstrument der 5-V-Gleichspannungsbereich (oder der nächst verfügbare höhere Bereich) eingestellt. Man sollte jetzt eine Ausgangsspannung von sehr genau 5 V messen. Der Wert darf minimal bei 4,75 V und maximal bei 5,25 V liegen, größere Abweichungen sind nicht zugelassen. Als Krönung des Ganzen gibt es im Bausatz noch eine Leuchtdiode und einen Widerstand. Die Leuchtdiode wird an die Stelle LED 1 eingelötet. Dabei muß man auf die Polung der Leuchtdiode achten. Das längere Bein ist der +-Anschluß.

Zusatzaufgaben als Anregung für Lehrer

1. Meßreihe. Die Ausgangsspannung in Abhängigkeit von der Ausgangslast, z. B. Lastströme von 100 mA, 1 A, 2 A, 3 A, die man durch unterschiedliche Widerstände oder mit einem Regelwiderstand erreichen kann.
Der Spannungsregler ist kurzschlußfest, daher kann nichts passieren. Die Widerstände können heiß werden, wenn sie zu wenig Leistung vertragen (5 V mit 3 A Last entspricht 15 W Leistung).
2. Meßreihe. Die Ausgangsspannung in Abhängigkeit von der Eingangsspannung. Am Wechselspannungseingang wird 2 V, 4 V, 5 V, 6 V, 7 V, 8 V, 9 V eingestellt und am Ausgang einmal ohne zusätzliche Last und einmal mit 1-A-Last gemessen.

Danach wird noch der Widerstand R1 eingelötet, der als Schutz für die Leuchtdiode dient. R1 hat den Wert 330 Ω. Die Farbringe zeigen dann orange-orange-braun-gold. Gold ist der sogenannte Toleranzring, der anzeigt, daß der Widerstand höchstens um 5% vom angegebenen Wert abweicht. Man kann auch 10%-Widerstände mit silbernem vierten Ring verwenden.

Wenn man die Leuchtdiode verkehrt herum eingelötet hat, so leuchtet sie nicht. Sie wird davon nicht zerstört, man sollte sie aber nicht zu lange in diesem Zustand lassen. Eine Steckleiste mit abgewinkelten Pfostensteckern „im 2,54-mm-Raster" wird am Schluß auf der Bestückungsseite mit den abgewinkelten Stiften eingesteckt und dann eingelötet. Wenn man keine passenden Stiftleisten bekommt, kann man auch längere einfach passend abschneiden oder aus kurzen eine lange konstruieren. Bei der POW5V sind nur 4 Stifte wirklich belegt, die anderen dienen der mechanischen Stabilität, wenn man die POW5V-Baugruppe später in ein Buchsenfeld steckt. Die Stiftleiste dient danach als Verbindung zu den restlichen Baugruppen des NDR-Klein-Computers. Den vollständigen Aufbau zeigt *Abb. 1.1.5*.

1.2 Die verwendeten Bauteile

Für alle, die sich mit Elektronik noch nicht so auskennen, sei hier eine kleine Kurzbeschreibung gegeben.

a) Widerstand

Widerstände dienen z. B. dazu, den Strom zu begrenzen. Der Stromfluß errechnet sich zu:

I = U/R. Dabei ist U die Spannung (in Volt, V) und R der Widerstand (in Ohm, Ω). Der Strom I besitzt dann die Einheit Ampere (A).

Der Widerstand wird in der Einheit Ohm gemessen, dabei sind dann 1000 Ohm = 1 kΩ und 1000 kΩ = 1 MΩ.

Abb. 1.2.1 zeigt das Schaltsymbol und die Bauform eines Widerstands. Neben der Widerstandsangabe wird bei Widerständen auch noch eine Leistungsangabe gemacht. Denn jeder Widerstand setzt Leistung in Wärme um und wird er zu heiß, so geht er kaputt. Die umgesetzte Leistung läßt sich auch berechnen und beträgt: P = U*U/R. Die Leistung wird in Watt gemessen.

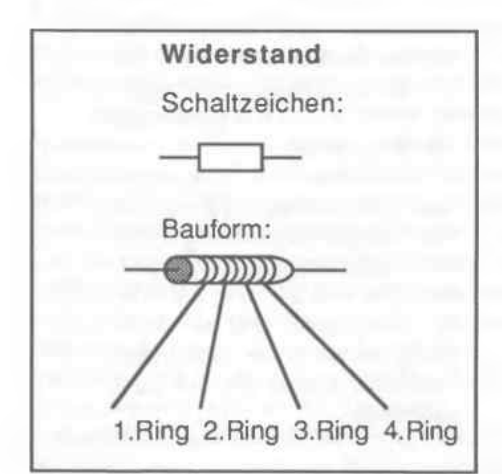


Abb. 1.2.1 Der Widerstand

Farbcode bei Widerständen:				
Farbe:	1.Ring 1. Ziffer	2.Ring 2. Ziffer	3.Ring Anzahl der Nullen	4.Ring Toleranz
schwarz	0	0	0	-
braun	1	1	1	-
rot	2	2	2	-
orange	3	3	3	-
gelb	4	4	4	-
grün	5	5	5	-
blau	6	6	6	-
violett	7	7	7	-
grau	8	8	8	-
weiß	9	9	9	-
gold			Wert * 0.1	+-5%
silber			Wert * 0.01	+-10%
keine Farbe				+-20%

Abb. 1.2.2 Farbcode bei Widerständen

Für unsere Schaltungen genügen Widerstände mit einer maximalen Leistung von ¼ bis ⅛ Watt, da die umgesetzten Energien sehr klein sind. Beispiel: Widerstand 330 Ohm, Spannung 5 V. Die umgesetzte Leistung beträgt dann 5 V*5 V/330 Ohm = 0.075 Watt. ⅛ Watt ist aber 0.125 W, also größer als die errechnete Leistung und damit kann man einen Widerstand mit ⅛ Watt einsetzen.

Der Widerstandswert wird nur selten auf die Widerstände in Klarschrift aufgedruckt, dazu sind sie viel zu klein. Man verwendet einen sogenannten Farbcode. *Abb. 1.2.2* zeigt die Farbcodetabelle. Normalerweise besitzen Widerstände vier Farbringe. Der vierte Ring ist meist etwas von den drei anderen Ringen entfernt, so daß man sie leicht identifizieren kann. Der erste Ring entspricht der ersten Ziffer des Widerstandswertes in Ohm. Der zweite Ring steht für die zweite Stelle. Der dritte Ring gibt die Anzahl der Nullen an, die man hinter die beiden Ziffern schreiben muß. Der vierte Ring schließlich gibt die Toleranz an. Damit wird festgelegt in welchem Bereich der Widerstandswert schwanken kann. So muß ein 330 Ohm Widerstand nicht exakt 330 Ohm besitzen, wenn seine Toleranz z. B. 10% beträgt, liegt der Widerstandswert zwischen 297 und 363 Ohm.

Beispiele:
braun schwarz rot gold: 1000 Ohm = 1 kΩ, 5%
orange orange braun gold: 330 Ohm, 5%
braun rot orange silber: 12 000 Ohm = 12 kΩ, 10%
braun schwarz blau gold: 10 000 000 Ohm = 10 MΩ, 5%

Manche Widerstände haben fünf Ringe. Dann geben die ersten drei Ringe die ersten drei Ziffern an, darauf folgt die Anzahl der Nullen und dann die Toleranz.

Den Toleranzring erkennt man auch daran, daß er meist eine goldene oder silberne Farbe hat. Widerstände mit 20% Toleranz, also ohne diesen Ring, sollten wir bei uns nicht einsetzen. 5% oder 10% sind verwendbar.

Für spezielle Aufgaben gibt es auch Widerstände mit 2% oder kleineren Toleranzen.

b) Kondensator

Kondensatoren können Ladungen speichern. Damit lassen sich unterschiedliche Aufgaben bewältigen. Bei der Spannungsversorgung dienen sie der Speicherung von Energie, um die Zeit zwischen den Energieschüben der Wechselspannung zu überbrücken. Kondensatoren haben aber auch noch andere Eigenschaften. So sind sie für Wechselspannungen durchlässig, lassen Gleichstrom aber nicht passieren. *Abb. 1.2.3* zeigt Schaltzeichen und Bauformen von Kondensatoren. Beim Kondensator werden zwei grundsätzliche Typen voneinander unterschieden. Der ungepolte und der gepolte Kondensator. Ungepolte Kondensatoren gibt es in sehr verschiedenen Bauformen, nur ein Teil davon ist hier dargestellt.

Die elektrische Größe beim Kondensator wird in Farad gemessen. Sie gibt sozusagen das Fassungsvermögen an. Da man normalerweise nur sehr kleine „Kapazitäten" verwendet, wird sie gerne in Mikrofarad (µF), Nanofarad (nF) oder Picofard (pF) angegeben. 1 F = 1000000 µF, 1 µF = 1000 nF, 1 nF = 1000 pF. Kondensatoren mit mehr als 1 µF sind meist gepolt. Das kommt daher, daß diese eine Trennschicht im Inneren des Kondensators verwenden, die polungsabhängig ist. Gepolte Kondensatoren gehen kaputt, wenn man sie falsch herum anschließt, da sich die innere Substanz zersetzt. Diese Polung ist keine grundsätzliche Eigenschaft von Kondensatoren, sondern gewissermaßen nur ein unerwünschter Nebeneffekt des Herstellungsverfahrens.

Bei den gepolten Kondensatoren gibt es zum Einen die sogenannten Elektrolyt-Kondensatoren und die Tantal-Kondensatoren.

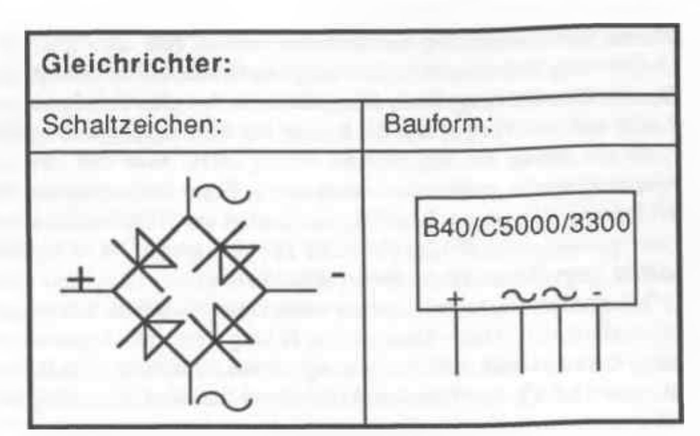
Die Tantal-Kondensatoren sehen aus wie kleine Perlen.

Beide Kondensatortypen sind meist an einer Stelle entweder mit dem Plus-Zeichen oder mit einem Minus-Zeichen beschriftet. Die Kondensatoren haben auch immer eine Maximal-Spannung, die man nicht überschreiten darf. Man sollte nur Kondensatoren mit ausreichender Spannungsfestigkeit verwenden.

Kondensator:	
Schaltzeichen:	Bauformen:
ungepolt	Schicht, Scheibe, Wickel
+ - gepolt	+ Tantal; + ELKO - Kerbe

Abb. 1.2.3 Der Kondensator

Abb. 1.2.4 Der Gleichrichter



Die unterschiedlichen Kondensator-Typen haben auch bei gleicher Kapazität abweichende elektrische Eigenschaften.

Bei den ungepolten sind die sogenannten Wickelkondensatoren in der Mikroelektronik nicht so beliebt. Im Gegensatz zu den Keramik-Kondensatoren (Scheibe), haben sie auch eine störende Induktivität (Spule), da sie aufgewickelt sind. Das kann zu Störungen führen. Die keramischen Kondensatoren werden gerne zum Entstören verwendet, sind aber stark temperaturabhängig. Die aufgedruckten Kapazitätswerte schwanken somit sehr stark. Für die Entstörung, so wie wir sie meist einsetzen, ist das unkritisch, aber als Frequenzerzeuger (z. B. CAS-Baugruppe) geht es nicht, dort muß man wieder Wickelkondensatoren oder noch besser Schichtkondensatoren einsetzen.

c) Gleichrichter

Die Eigenschaften des Gleichrichters haben wir schon im Abschnitt 1.1 kennengelernt. Im Inneren eines Gleichrichters verbergen sich 4 Dioden, die wie aus dem Schaltzeichen deutlich wird, miteinander verdrahtet sind *(Abb. 1.2.4)*. Der Gleichrichter wird in sehr verschiedenen Formen geliefert. Ein Beispiel ist dargestellt. Zwei Bauformen können auf die Leiterplatte gesteckt werden. Man muß beim Gleichrichter immer auf die aufgedruckte Anschlußbelegung achten, bevor man ihn einsetzt.

Er besitzt zwei Wechselspannungseingänge und einen Plus- sowie einen Minus-Ausgang. Beim Gleichrichter sind zwei Angaben wichtig. Die Spannung, die er maximal verträgt und der Strom, der durch ihn fließen darf. Die Bezeichnung B40/C5000/3300 bedeutet: Brückengleichrichter mit maximal 40 Volt Eingangsspannung, mit 5000 mA (also 5 A) maximalem Strom bei Kühlung durch eine Kühlschelle und mit 3300 mA (also 3,3 A) maximalem Strom bei Luftkühlung. Die Strombegrenzung kommt daher, daß auch im Gleichrichter eine Leistung umgesetzt wird.

d) Leuchtdiode

Jeder kennt Glühbirnen und weiß, daß sie neben Licht auch beachtliche Wärme erzeugen. Es gibt aber auch kaltes Licht, z. B. bei einer Leuchtstoffröhre. Noch besser ist die Lichtumsetzung bei speziellen Halbleitermaterialien, z. B. dem sogenannten Galliumsarsenid. Die Verlustenergie, wie sie normalerweise entsteht, wird zum allergrößten Teil in Licht umgesetzt, der Rest wird in

Wärme verwandelt. Die Leuchtdiode leuchtet also nur, wenn man sie in der sogenannten Flußrichtung betreibt. *Abb. 1.2.5* zeigt ein Schema. Das Dioden-Symbol haben wir schon beim Gleichrichter kennengelernt. Eine Diode besitzt eine Anode und eine Katode. Wenn man die Anode mit dem Pluspol und die Katode mit dem Minuspol einer Spannungsquelle verbindet, so fließt ein Strom, bei umgekehrter Polung nicht. Man darf eine Diode aber nie direkt an die Spannungsquelle anschließen, sonst ist der Stromfluß unbegrenzt. Daher verwendet man einen Widerstand. Für unsere Leuchtdioden genügt ein Widerstand von ca. 330 Ohm. Damit wird bei einer Spannung von 5 V der Strom auf 15 mA begrenzt. Da an der Diode aber auch eine Spannung abfällt, liegt der wirkliche Strom dann darunter.

Leuchtdioden nehmen normalerweise keinen Schaden, wenn man sie falsch polt, sie leuchten nur einfach nicht. Daher kann man die Polung auch einfach ausprobieren, denn Leuchtdioden sind nicht bedruckt und manchmal ist es schwer, die Polung so herauszubekommen. Eine kleine Batterie (4.5 V), ein Widerstand (330 Ohm) und die Leuchtdiode helfen da weiter. Man schaltet alles in Reihe und wenn die Leuchtdiode leuchtet, so kann man die Polung notieren.

e) Spannungsregler

Ein Spannungsregler ist eine komplizierte integrierte Schaltung, die aus Transistoren und Widerständen besteht. Um das Innere brauchen wir uns zunächst nicht zu kümmern, wichtig ist die richtige Anschlußbelegung.

Dabei sollte man sich ein Datenblatt vom Hersteller besorgen, da die Anschlußbelegung u. U. abweichend sein kann. Für den IC-Typ 78H05 oder TBA0123 ist sie aber durch das Layout unsere Leiterplatte vorgegeben und man kann das IC gar nicht falsch einbauen.

Bei Spannungsreglern interessieren eigentlich drei wesentliche Angaben.

1. Welche maximale Spannung kann der Regler am Eingang vertragen?
   Diese Angabe ist wichtig, wenn man am Eingang eine sehr hohe Spannung im Verhältnis zur Ausgangsspannung verwendet. Das ist aber normalerweise nicht erwünscht, da bei hohem Unterschied auch eine große Verlustleistung im Regler in Wärme umgesetzt wird.
2. Welche Ausgangsspannung liefert der Regler?
   Neben dem 5-V-Regler gibt es auch welche für andere feste Spannungen, wie 12 V, – 5 V oder – 12 V oder solche für variable Ausgangsspannung.
3. Welchen Strom kann der Regler maximal liefern?

Der Regler 78H05 ist für 5 A ausgelegt, der TBA 0123 für 3 A.

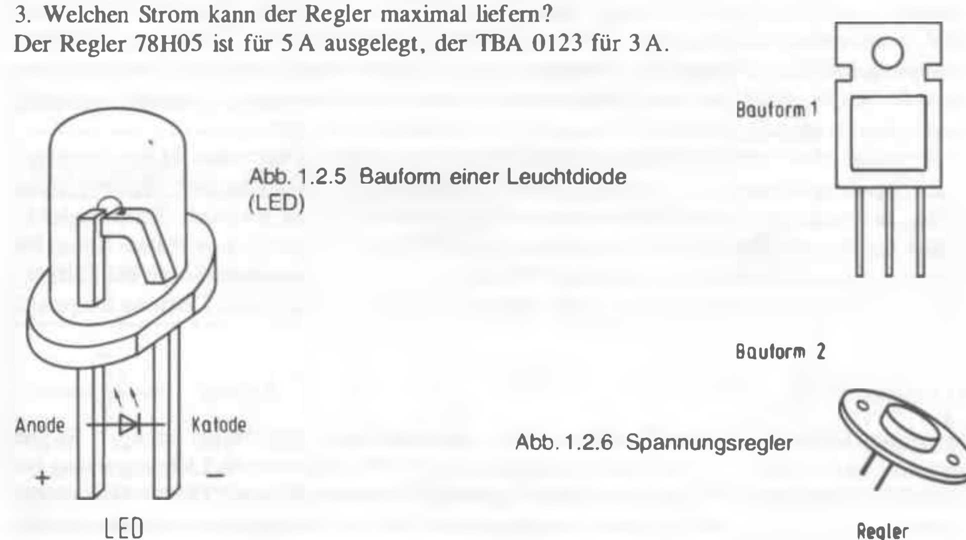


Abb. 1.2.5 Bauform einer Leuchtdiode (LED)

Abb. 1.2.6 Spannungsregler

Der Strom hängt vom Reglertyp ab und ist nur aus dem dazugehörigen Datenbuch zu entnehmen, das man von der Herstellerfirma der Regler bekommen kann.

Schließlich ist noch die umgesetzte Leistung von Bedeutung. Sie errechnet sich aus (Eingangsspannung – Ausgangsspannung)*Ausgangsstrom.
Beispiel:
10 V Eingangsspannung, 5 V Ausgangsspannung, 3 A Ausgangsstrom, also
P = (10 V − 5 V) · 3 A = 15 Watt.

Diese Leistung wird in Wärme umgesetzt und muß abgeführt werden. Verwendet man keinen Kühlkörper, so kann die Temperatur des Spannungsreglers stark ansteigen und ab ca. 70 °C schaltet er ab.

Der Kühlkörper sorgt dafür, daß die Wärme an die Umgebungsluft abgegeben wird. In ganz extremen Fällen kann man einen kleinen Ventilator verwenden, der die Wärme abtransportiert.

1.3 Andere Spannungsquellen

Heute gibt es schon eine Vielzahl fertiger Mikrocomputernetzteile, die alle wichtigen Spannungen von 5 V, + 12 V, − 12 V liefern können. Wer mehr mit Mikrorechnern machen will, sollte vom Selbstbau absehen und gleich ein solches Netzteil besorgen. Dabei sollte der Strom bei + 5 V mit ca. 6 A gegeben sein, bei + 12 V mit ca. 2 A (wenn man Floppys anschließen will) und bei − 12 V genügt 1 A. Schön ist es, wenn man auch noch 26 V bekommt, sie benötigt man später, wenn man EPROMs programmieren will.

2 Kurze Einführung in die Digitaltechnik

Die Digitaltechnik vollständig abzuhandeln, würde den Rahmen des Buches sicher sprengen. Dennoch soll hier versucht werden, wenigstens einen kurzen Überblick zu vermitteln.

Im Jahr 1941 wurde in Deutschland die erste programmgesteuerte, elektronische Rechenanlage in Betrieb genommen. Das Gerät bestand aus 2600 höchst sinnvoll verschalteten Relais. Erdacht hatte diesen ersten Computer der Welt, der auch wirklich vollständig funktionierte, ein Mann namens Konrad Zuse. Er wollte damit die langwierigen und auch fehlerträchtigen Berechnungen im Bauingenieurwesen und auch anderswo automatisieren und damit sicherer und schneller durchführbar machen. Der Computer trug den Namen Z3. Er ist im deutschen Museum in München zu besichtigen.

2.1 Digitale Signale (Dr. Hans Hehl)

Die Z3 von Konrad Zuse war ein erstaunliches Gerät. Denn sie konnte neben den Grundrechenarten zum Beispiel auch Wurzeln ziehen. Das ist deshalb so erstaunlich, weil die Z3 dazu (natürlich in elementarer Form) alle Merkmale programmgesteuerter Universalrechenmachinen aufweisen mußte, wie sie auch heute noch gültig sind. Erfunden und in die Maschine eingebaut hat das alles im wesentlichen ein einziger Mann, eben Konrad Zuse, der dafür in den Jahren nach 1970 auch vielfältig geehrt wurde. Eine seiner wichtigen Ideen beim Bau der Z3 war die Verwendung eines Zahlensystemes, das besser an Maschinen angepaßt ist als unser gewöhnliches Zehnersystem. Intern rechnete die Z3 mit den sogenannten Dualzahlen. Weshalb das so gut war, soll gleich erklärt werden: Es hängt mit der Verwendung von Relais zusammen.

Ein Relais ist ja nichts weiter als ein elektrischer Schalter, der mit elektrischem Strom ein- und ausgeschaltet werden kann. Bei einem Relais macht also der Strom das, was man bei einer Taschenlampe mit dem Daumen von Hand machen muß: Man schaltet den Schalter der Lampe ein oder aus. Entsprechend wird die Lampe leuchten oder nicht. Eine Verbindung zu den Zahlen kann man schlagen, wenn man verabredet, daß zum Beispiel der Zustand der Taschenlampe Null sein soll, wenn sie ausgeschaltet ist und Eins, wenn sie eingeschaltet ist. So merkwürdig künstlich und willkürlich so eine Verabredung zunächst erscheint, die ganze Computerindustrie ist in gewissem Sinn darauf aufgebaut.

Ein Relais, eine Taschenlampe, überhaupt ein physikalisches Gerät, das zwei Zustände annehmen kann, von welchen man verabreden kann, daß der eine Zustand Null, der andere Eins bedeuten soll, das sind Beispiele für die Realisierung einer sogenannten binären Variablen.

Einer Taschenlampe sieht man nicht an, mit welcher Spannung die Glühbirne betrieben wird. Allerdings weiß man, daß sicher nicht so hohe Spannungen wie z. B. 220 V verwendet werden. Genauso sind die Spannungswerte (Pegel) bei einer elektrisch dargestellten binären Variablen (ein- oder ausgeschaltet) prinzipiell nicht vorgegeben. Sie hängen jeweils von der technischen Konzeption des Gerätes ab. Dem Binärwert 0 kann man zum Beispiel die Spannung 0 V ebenso zuordnen wie die Spannung − 12 V. Allgemein spricht man von einem L-Pegel (Low), wenn der Pegelwert näher bei „minus unendlich“ liegt und von einem H-Pegel (High), wenn der Pegelwert

näher bei „plus unendlich" liegt. In der Praxis wird meist dem L-Pegel der Wert 0 der binären Variable und dem H-Pegel der Wert 1 zugeordnet (positive Logik).

Dualzahlen binär dargestellt

Wenn man begreifen will, wie Zahlen in einem Computer dargestellt werden, dann ist es zum Beispiel günstig, sich den Kilometerzähler eines Autos vorzustellen. Es sei ein Modell, das keine Hundert-Meter-Einteilung besitzt. Dann wird dort beim Fahren das Anzeigerad für die einzelnen Kilometer, also das Rad ganz rechts, von einer mit den Auto-Rädern verbundenen Welle gedreht und zeigt nacheinander 0, 1, 2, 3, 4, . . . Auf diesem Anzeigerad (und auf den anderen weiter links auch) befinden sich zehn Ziffern, die der Reihe nach gezeigt werden. Immer dann, wenn das Einerrad eine Umdrehung vollendet, also beim Umschalten von 9 auf 0, wird das benachbarte weiter links befindliche Rad um eine Ziffer weitergedreht. Stand es vorher auf 0, dann steht es nach einer solchen Situation auf 1. Dazu besitzt das Einerrad einen Mitnehmer, der das Zehnerrad dann mitnimmt. Also nach zehn gefahrenen Kilometern steht tatsächlich auch 10 auf dem Kilometerzähler. Das Zehnerrad zählt also mit, wie oft das Einerrad sich gedreht hat und merkt so an, wievielmal zehn Kilometer zurückgelegt wurden. Das Zehnerrad selbst besitzt ebenfalls einen Mitnehmer, der bei Vollendung einer Umdrehung das benachbarte Hunderterrad um eins weiterdreht. Und dieses Hunderterrad wiederum kann das Tausenderrad mitnehmen, was selbst wieder das Zehntausenderrad mitnimmt. Und so weiter.

Daß die Anzeigeräder jeweils 10 Ziffern tragen, das rührt von unserer Gewohnheit her, im Zehnersystem zu rechnen. Eine ziemlich merkwürdige, aber durchaus mögliche Konstruktion eines solchen Kilometerzählers könnte darin bestehen, daß man auf die Anzeigeräder nur auf der einen Seite des Umfanges 0 und auf der anderen Hälfte 1 anschreibt und den Mitnahmemechanismus so gestaltet, daß beim Drehen von 1 auf 0 das weiter links befindliche Rad um eine halbe Umdrehung weiter gedreht wird. Das Einerrad zählt dann von 0 bis 1. Links daneben befindet sich das Zweierrad, das um eins weiter gedreht wird, wenn das Einerrad einmal ganz herum kommt und dabei der zweite Kilometer abgefahren wird. Nach zwei gefahrenen Kilometern steht dann 10 auf diesem merkwürdigen Zähler. Auch das Zweierrad dreht beim Übergang von 1 auf 0 ein weiter links befindliches Rad. Es sind hier einfach einmal für einen vierstelligen Zähler mit der verrückten Zweiereinteilung die Anzeigestellungen und die gefahrenen Kilometer aufgezählt:

```
0000  =   0
0001  =   1
0010  =   2
0011  =   3
0100  =   4
0101  =   5
0110  =   6
0111  =   7
1000  =   8
1001  =   9
1010  =  10
1011  =  11
1100  =  12
1101  =  13
1110  =  14
1111  =  15
```

An dieser Aufstellung kann man, wenn man scharf hinschaut, erkennen, daß in einer Kolonne von oben nach unten (links vom Gleichheitszeichen) immer nur die Ziffern 0 auf 1 auftauchen. Man könnte also für jede der vier Stellen eine binäre Variable hernehmen und diese vier Variablen dann jeweils so schalten, wie es das Muster aus Nullen und Einsen verlangt, das gerade auf dem Kilometerzähler erscheint. Zum Beispiel leuchtet bei vier nebeneinanderliegenden Taschenlampen genau die ganz rechts außen liegende. Dann kann man sagen, daß damit die Zahl Eins binär dargestellt ist. Wenn alle vier Lampen eingeschaltet sind, dann ist damit die Zahl 15 binär dargestellt. Jeder Kombination von „Ein" und „Aus" entspricht also ganz natürlich eine Zahl.

Mathematiker nennen Gebilde, wie sie in der linken Spalte auftauchen, Dualzahlen, wenn sie betonen wollen, daß sie in einem System arbeiten, das nur die Ziffern 0 und 1 benutzt. Wie eben gesagt, kann man also die Dualzahlen zum Beispiel mit einer geeigneten Anzahl von Taschenlampen binär darstellen. Zuse benutzte in seiner Z3 Relais, um damit Dualzahlen binär in seiner Maschine darzustellen.

Noch etwas Theorie

Ein Zahlensystem mit den beiden Ziffern 0 und 1 unterscheidet sich von unserem gewohnten Zehnersystem durch einen wesentlich kleineren Abstand der Stellenwerte. Was besagt dies aber? Grundsätzlich können wir je nach Art der Anordnung von Zeichen für Zahlen Additionssysteme und Positionssysteme unterscheiden.

Ein Additionssystem ist zum Beispiel das römische Zahlensystem. In ihm wird das Jahr 1768 als MDCCLXVIII dargestellt. Die eigentliche Zahl ergibt sich durch Addition der einzelnen Zahlzeichen. Das heutzutage benützte Positionssystem (auch Stellenwertsystem genannt) wertet dagegen die Stellung des Zahlzeichens mit aus.
Ein Beispiel soll dies verdeutlichen: In dem römischen Zeichen III für die Zahl Drei hat jede der drei Ziffern den Zahlwert 1 und die Zahl ergibt sich durch die Addition der drei einzelnen Zahlenwerte.

Im dekadischen Positionssystem ergibt die Zeichenanordnung 111 die Zahl Einhundertelf. Alle Zeichen haben den gleichen Zahlwert 1, aber einen unterschiedlichen Stellenwert. Der niedrigste

Tabelle 2.1.1 Dezimalzahl und Einschaltkombination

Dezimal-zahl	Kombination
1	0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 1 0
3	0 0 0 0 0 0 1 1
4	0 0 0 0 0 1 0 0
5	0 0 0 0 0 1 0 1
6	0 0 0 0 0 1 1 0
7	0 0 0 0 0 1 1 1
8	0 0 0 0 1 0 0 0
9	0 0 0 0 1 0 0 1
10	0 0 0 0 1 0 1 0
:	:
31	0 0 0 1 1 1 1 1
:	:
255	1 1 1 1 1 1 1 1

Stellenwert (Einer) steht ganz rechts, dann folgen Zehner und Hunderter. Jeder Stellenwert beträgt $\frac{1}{10}$ des links von ihm stehenden.

Beim Binärsystem sind die Stellenwerte die Potenzen der Zahl (Basis) 2, also die Zahlen 1, 2, 4, 8, 16, 32, 64, . . . usw.

Acht Taschenlampen oder ein Byte?

Wir verwenden nun acht Taschenlampen, die wir in eine Reihe legen und beliebig ein- bzw. ausschalten. Damit ergeben sich $2^8 = 256$ Einschaltkombinationen. Wenn die Zuordnung dieser Kombinationen zu den Zahlen 0 bis 255 mit der Rechenregel „Dezimalzahl ergibt sich durch Aufsummieren der Zweierpotenzen" durchgeführt wird, dann ergibt sich das Schema wie beim Kilometerzähler. In *Tabelle 2.1.1* sind einige Dezimalzahlen und die entsprechenden Kombinationen der Werte 0 und 1 aufgeführt (eingeschaltet = 1, ausgeschaltet = 0), die sich beim Zählen wie vorhin ergeben würden.

Die Zuordnung kann nun überprüft werden. Die Kombination 0 0 0 1 1 1 1 1 ergibt als Summe der Zweierpotenzen die Zahl 31.

$$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
$$0 + 0 + 0 + 16 + 8 + 4 + 2 + 1$$

Umgekehrt kann aus einer Dezimalzahl zwischen 0 und 255 die zugehörige Dualzahl ermittelt werden, indem fortlaufend die Zweierpotenzen, beginnend bei 2^7, von der Zahl bzw. vom übrig bleibenden Rest abgezogen werden. Würde die Differenz negativ, so wird eine Null aufgeschrieben und die nächst kleinere Zweierpotenz verwendet. Ist die Differenz positiv, so wird eine 1 aufgeschrieben und mit dem Rest weiter gearbeitet. Probieren wir dies mit der Zahl 18 aus.

Tabelle 2.1.2 Gegenüberstellung der drei Stellenwertsysteme

Dezimal	Sedezimal	Dual
0	0	0000 0000
1	1	0000 0001
2	2	0000 0010
3	3	0000 0011
⋮	⋮	⋮
8	8	0000 1000
9	9	0000 1001
10	A	0000 1010
11	B	0000 1001
12	C	0000 1100
13	D	0000 1101
14	E	0000 1110
15	F	0000 1111
16	10	0001 0000
17	11	0001 0001
⋮	⋮	⋮
94	5E	0101 1110
⋮	⋮	⋮
171	AB	1010 1011
⋮	⋮	⋮
255	FF	1111 1111

18 − 128 = ? geht nicht, also 0
18 − 64 = ? geht nicht, also 0
18 − 32 = ? geht nicht, also 0
18 − 16 = 2 geht , also 1
2 − 8 = ? geht nicht, also 0
2 − 4 = ? geht nicht, also 0
2 − 2 = 0 geht , also 1
0 − 1 = ? geht nicht, also 0

Die Kombination für die Zahl 18 lautet also von oben nach unten: 0 0 0 1 0 0 1 0.

Eine solche Kombination von Nullen und Einsen nennt man „Byte“. Dieses Byte besteht aus acht Bits. Ein Bit, abgeleitet von „binary digit“, ist die kleinste Darstellungseinheit für Binärdaten. Ein Bit repräsentiert eine Binärstelle in einem Byte.

Für Umwandlungsübungen seien noch einige Beispiele angegeben.

85 = 0 1 0 1 0 1 0 1
138 = 1 0 0 0 1 0 1 0
255 = 1 1 1 1 1 1 1 1

2.2 Der Treiber und Logikschaltungen (Dr. Hans Hehl)

Vom Relais zum Transistor

Genug der Mathematik, nun sei diskutiert, wie das Ein- und Ausschalten der Glühbirnchen unserer acht Taschenlampen automatisiert werden kann. 1941 verwendete K. Zuse dazu Relais. Aber so ein Relais kann nicht Hunderte von Schaltvorgängen pro Sekunde durchführen, die Kontakte sind dazu zu träge. Es ist schon eigenartig, daß nur 7 Jahre später, also 1948 ein Ersatz für das langsame Relais entdeckt wurde. Die Amerikaner Bardeen, Brattain und Shockley entdeckten den Transistoreffekt an einem Germaniumkristall und erhielten dafür 1956 den Nobelpreis für Physik.

Mit dem Transistor stand ein Bauteil zur Verfügung, das keine mechanischen Teile enthält, sehr schnell schalten kann und weniger Strom als der Elektromagnet eines Relais benötigt. *Abb. 2.2.1* zeigt den Schaltplan eines Transistorschalters.

So eine Schaltung wird zum Beispiel benötigt, wenn ein Computer Lampen, Motoren usw. schalten soll, um also eine Verbindung zur Außenwelt zu schaffen. Bevor wir die Teile der Schaltung näher betrachten, müssen wir uns jedoch dem Problem der Stromrichtung zuwenden.

Man hatte vor der Zeit der Elektronenröhren und der Halbleiter einfach die Stromrichtung vom Pluspol der Spannungsquelle über den Verbraucher zum Minuspol festgesetzt (technische Stromrichtung). Die normalen Träger der Elektrizität, die Elektronen, fließen aber vom Minuspol über den Verbraucher zum Pluspol, wie man erst später entdeckte. Wir verwenden hier diese Elektronenflußrichtung.

Wichtigster Teil der Treiberschaltung nach *Abb. 2.2.1* ist der Transistor (BC-107). Er besteht im Inneren aus drei Halbleiterschichten mit wechselnder Leitfähigkeit. Halbleiter, zum Beispiel Silizium oder Germanium, leiten den Strom schlechter als Metalle. Werden geringste Mengen eines anderen Metalles (z. B. Antimon) hinzugefügt, verändert sich die Leitfähigkeit der Schicht erheblich. Verfolgen wir nun den Elektronenfluß durch den Transistor. Vom Minuspol der Batterie fließen die Elektronen zum Emitter E (durch eine Pfeilspitze gekennzeichnet, die aus dem Kreissymbol heraus zeigt, npn-Typ). Wieviel Elektronen nun zum Collektor C (bzw. Kollektor) und damit durch die Lampe L1 fließen können, hängt vom Stromfluß am Steuereingang (Basis) B ab, der vom Emitter über Basis, Widerstand R, geschlossenen Schalter S zum Pluspol der Batterie fließt. Ein geringer Emitter-Basis-Strom bewirkt einen großen Emitter-Kollektor-Strom, man spricht von einer „Stromverstärkung“. Großer Strom bedeutet aber nach dem Ohmschen Gesetz einen kleinen Widerstand der Emitter-Kollektor-Strecke, der Transistor „schaltet durch“. Die Stromverstärkung beträgt einige „Hundertfache“.

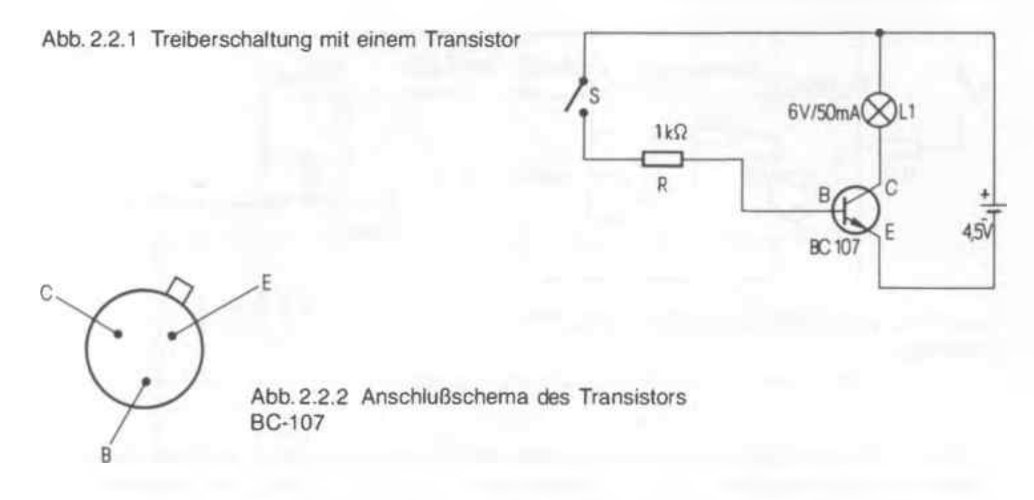


Abb. 2.2.1 Treiberschaltung mit einem Transistor

Abb. 2.2.2 Anschlußschema des Transistors BC-107

Nach der Theorie zur Praxis. Verwendet werden kann der Transistor BC-107 oder ein vergleichbarer npn-Typ (wie z. B. BC-109 und andere). Die Kennzeichnung der drei Anschlußdrähte des Transistors ergibt sich aus *Abb. 2.2.2*. Der Draht dicht neben dem Gehäusevorsprung ist der Emitter, wobei man den Transistor von unten, also von der Anschlußdrahtseite her anschaut.

Achtung: wird der Basis-Widerstand R überbrückt, liegt die Spannung der Batterie voll an Emitter und Basis an, ein großer Strom fließt und eine Zuleitung im Transistor schmilzt durch.

Löten: Übung macht den Meister

Wenn Sie die im Buch angegebenen Schaltungen aufbauen wollen, dann verwenden Sie bitte einen kleinen Lötkolben mit etwa 20 W Leistung und eine feine Dauerlötspitze, die nicht verzundert. Reine Kupferspitzen sind weniger geeignet. Die Spitze reinigt man vor jedem Lötvorgang mit einem feuchten Spezialschwämmchen oder mit einem Baumwollappen. Als Lot wird ein 1 mm dünner Lötdraht verwendet, der im Inneren ein Flußmittel auf Harzbasis enthält. Säurehaltige Flußmittel wie Lötfette oder sogar Salzsäure dürfen auf gar keinen Fall verwendet werden, da die Säurereste eine Korrosion der Leiterbahnen und Bauteile bewirken. Zum Löten erwärmen Sie mit der Lötkolbenspitze solange die zu verbindenden Teile, bis das gleichzeitig an die Teile gehaltene Lot schmilzt und die Teile überzieht. Die Lötstelle darf bis zum Erkalten nicht bewegt werden. Eine gute Lötstelle verbindet die Bauteile mit nur wenig Lötzinn, das eine hellglänzende Oberfläche besitzt. Üben Sie dies nicht mit Ihren Bausatzplatinen, sondern an versilberten Schaltdrahtresten oder Kupferlitze. Alles Handwerkszeug und das Lötzinn sollten Sie im Elektronikfachhandel kaufen, damit Sie sicher sind, daß Ihre Arbeitsmittel auch geeignet sind.

Treiber mal zwei

Wir benötigen zu Versuchszwecken wieder unsere Treiberschaltung aus dem ersten Abschnitt, *Abb. 2.2.1*. Die Lampe leuchtete, wenn der Schalter geschlossen wurde.

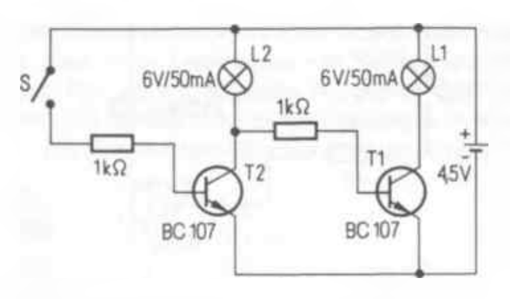


Abb. 2.2.3 NICHT-Glied: Doppelte Treiberschaltung

Abb. 2.2.4 Schaltsymbol des NICHT-Gliedes

Diese Treiberschaltung sei jetzt erweitert. Anstelle des Schalters wurde nochmals die gleiche Treiberschaltung eingesetzt. *Abb. 2.2.3* zeigt die neue Schaltung. Nach dem Anschließen der Batterie leuchtet Lampe L1. Wird aber der Schalter S geschlossen, so erlischt Lampe L1 und L2 leuchtet. Warum?

Lampe L1 leuchtet zunächst, da ein kleiner Steuerstrom vom Transistor T1 über Widerstand und Glühfaden von L2 fließen kann, ohne daß L2 leuchtet. Da der Schalter offen ist, kann kein Steuerstrom bei Transistor T2 und damit auch kein Strom von dessen Emitter zum Kollektor fließen. Der Transistor T2 besitzt in diesem Zustand keinen Einfluß auf die Schaltung. Schließen wir aber den Schalter, so bewirkt der entstehende Steuerstrom einen großen Stromfluß durch T2 und die Lampe L2. Wir könnten auch Emitter und Kollektor von T2 mit einem Draht überbrücken, denn großer Stromfluß bedeutet kleinen Widerstand (Spannung konstant). Jetzt fließt kein Steuerstrom mehr durch T1, weil die Spannung am Kollektor von T2 fast ganz auf Null abgesunken ist. L1 erlischt deshalb.

Wir bezeichnen nun den Schalter als Eingang, dessen offenen Schalterzustand mit der Zahl 0 und den geschlossenen mit 1. Die Lampe L1 wird zum Ausgang erklärt. Den Leuchtzustand kennzeichnen wir mit der Zahl 1. Dann erhalten wir folgenden Zusammenhang:

Schalter	Lampe L1
0	1
1	0

Das Eingangssignal erscheint am Ausgang invertiert, also genau umgekehrt. So eine invertierende Schaltung wird als NICHT-Glied bezeichnet. *Abb. 2.2.4* zeigt das Schaltsymbol der invertierenden Schaltung.

Nicht nur NICHT

Ein Gesichtspunkt ist besonders interessant. Man kann das Verhalten einer solchen Schaltung, wie die des Inverters, einerseits am konkreten Objekt studieren und andererseits das Wesentliche daran, daß nämlich ein Null-Zustand an der Eingabe in einen Eins-Zustand an der Ausgabe verwandelt wird und ein Eins-Zustand an der Eingabe in einen Null-Zustand an der Ausgabe, in einer Tabelle ganz kurz und trocken notieren. *Abb. 2.2.5* zeigt einfach an, was welchem Eingabewert an der Ausgabe durch die verwendete Schaltung zugeordnet wird. Der Inverter hatte an seiner Eingabe nur eine binäre Variable, den einen Schalter. Es gibt nun Schaltungen (die

Neue Norm	Alte Norm	Beispiel	Wahrheitstafel	
E — 1 — A		7404	E A 0 1 1 0	Nicht-Glied
E E — & — A		7408	E E A 0 0 0 0 1 0 1 0 0 1 1 1	Und-Glied
E E — ≥1 — A		7432	E E A 0 0 0 0 1 1 1 0 1 1 1 1	Oder-Glied
E E — =1 — A	≠	7486	E E A 0 0 0 0 1 1 1 0 1 1 1 0	Exclusiv-Oder-Glied
E E — & — A		7400	E E A 0 0 1 0 1 1 1 0 1 1 1 0	Nand-Glied-(Nicht-Und)
E E — ≥1 — A		7402	E E A 0 0 1 0 1 0 1 0 0 1 1 0	Nor-Glied-(Nicht-Oder)

Abb. 2.2.5 Einige Logikglieder und ihre Wahrheitstafeln

neben dem Inverter eine der Grundlagen der Computerei überhaupt bilden), die zwei oder mehrere Eingänge haben und die jeder Eingangskombination von Nullen und Einsen genau ein zugehöriges Ergebnis (0 oder 1) am Ausgang zuordnen. Zum Beispiel gibt es Schaltungen mit zwei Eingängen und einem Ausgang, wo genau dann eine 1 am Ausgang erscheint, wenn der eine Eingang UND der andere Eingang den Zustand 1 besitzen; in allen anderen Fällen erscheint eine Null am Ausgang. Eine solche Schaltung, sie kann in vielen Varianten aufgebaut werden, heißt ihrem Verhalten entsprechend UND-Schaltung oder in der Fachsprache der Digitaltechniker UND-Glied. *Abb. 2.2.5* zeigt sowohl die Zuordnungstabelle, die zum UND-Glied gehört, als auch seine abstrakten Schaltsymbole nach alter und neuer Norm. Neben diesem digitalen Schalt-Glied werden noch andere wichtige aufgeführt, deren Verhalten aus der zugehörigen Zuordnungstabelle abgelesen werden kann. Die Namen solcher Schaltglieder sind vom Verhalten abgeleitet. Solche Schalt-Glieder mit mehreren Eingängen heißen auch „Verknüpfungen", weil sie die Zustände der Eingänge hernehmen und zu einem Ausgangssignal verknüpfen.

Die wichtigsten drei Logikglieder sind das NICHT-, UND- sowie das ODER-Glied. Aus diesen Grundschaltungen lassen sich alle anderen Logikglieder zusammensetzen. So entsteht das NICHT-UND-Glied durch eine Reihenschaltung der Einzelglieder UND und NICHT. Mit diesen Logikgliedern und ihren Kombinationen werden im Computer Rechenvorgänge wie Addition und Subtraktion durchgeführt und Zahlen gespeichert. Wir wollen nun zwei Einzelglieder näher betrachten, das UND-Glied und das Exklusiv-ODER-Glied. Beim UND-Glied besitzt der Ausgang nur dann den Zustand 1, wenn alle Eingänge den Zustand 1 besitzen (es können mehrere Eingänge vorhanden sein). Beim Exklusiv-ODER-Glied besitzt der Ausgang nur dann den Zustand 1, wenn nur einer der vorhandenen Eingänge den Zustand 1 hat.

Wieviel ist eins und eins?

Für binäre Rechenvorgänge, wobei meist aus zwei binär dargestellten Zahlen durch Verknüpfungen eine neue entsteht, gibt es bestimmte Regeln, von denen wir uns kurz die für die binäre Addition anschauen. Zwei binäre Zahlen werden addiert, indem, mit dem niedrigsten Stellenwert beginnend, jedes Bit mit dem gleichwertigen Bit der anderen Zahl nach folgenden Regeln addiert wird:

0 + 0 ergibt 0
0 + 1 ergibt 1
1 + 0 ergibt 1
1 + 1 ergibt 0,

aber mit einem Übertrag 1.

Dieser Übertrag wird immer zum nächsthöheren Stellenwert addiert. Wir addieren die Zahlen 00111111 (63) und 00100110 (38) und beginnen mit den Ziffern ganz rechts. Das schaut dann so aus.

Dezimalzahl	Binärzahl	
63	0 0 1 1 1 1 1 1	
+ 38	0 0 1 0 0 1 1 0	
11	1 1 1 1 1	Übertrag
101	0 1 1 0 0 1 0 1	

Es sei nun überlegt, wie man mit Logikgliedern eine solche Addition verwirklichen kann.

Vergleicht man die Rechenregeln mit den Wahrheitstafeln in *Abb. 2.2.5*, so entsprechen diese der Verknüpfung eines Exklusiv-ODER-Gliedes, das die Summe zweier Bits bildet, mit einem UND-Glied, das den Übertrag ermittelt. Für jeden weiteren Stellenwert benötigt man aber den vorhergehenden Übertrag, der mit der Summe verknüpft werden muß. Um zwei Bits zu addieren, brauchen wir also vier UND-, zwei NICHT- und drei ODER-Glieder. Für ein Byte brauchen wir dann acht solcher Logikgruppen.

Hexerei oder sedezimale Zahlen

Um nicht immer mit den langen Kolonnen aus Nullen und Einsen bei Binärzahlen arbeiten zu müssen, gibt es zur Vereinfachung der Zahlendarstellung das Sedezimal-System (Sechzehnersystem oder fälschlicherweise Hexadezimalsystem genannt). Dieses benutzt 16 Zeichen (die Ziffern 0–9 und die Buchstaben A–F) im Gegensatz zum Dezimalsystem, das 10 Ziffern (0–9) oder das Binärsystem, das nur zwei Ziffern (0 und 1) verwendet.

Die *Tabelle 2.1.2* des letzten Abschnitts zeigt eine Gegenüberstellung der drei Stellenwertsysteme. Eine achtstellige Binärzahl ergibt eine nur zweistellige Sedezimalzahl.

Die Umwandlung einer Binärzahl in eine Sedezimalzahl ist einfach:

Man schreibt unter die Binärzahl von rechts beginnend für je vier Bit den Stellenwert, also die Potenzen der Zahl 2, und beginnt beim fünften Bit von vorne. Das sieht bei der Dezimalzahl 83 so aus:

0 1 0 1   0 0 1 1
8 4 2 1   8 4 2 1

Nun addiert man bei jeder Bit-Vierergruppe die Stellenwerte der Bits mit dem Wert 1, das ergibt die Zahlen 5 und 3. Die Sedezimalzahl lautet also 53. Bei der Dezimalzahl 165 ergibt sich:

1 0 1 0   0 1 0 1
8 4 2 1   8 4 2 1
10   5

Die Zahl 10 muß in das sedezimale System umgewandelt werden, ergibt also nach *Tabelle 2.1.2* den Buchstaben A. Die Sedezimalzahl lautet also A5.

Umgekehrt läßt sich eine Sedezimalzahl leicht in eine Binärzahl umwandeln, da nur jede Stelle der Sedezimalzahl durch die dazugehörige Bitkombination ersetzt werden muß. Buchstaben muß man vorher in die Dezimalzahl verwandeln, und dann setzt man unter den in zwei Vierergruppen angeschriebenen Stellenwerten die Bits der Summanden jeder Zahl auf den Wert 1. Die Sedezimalzahl 4 D ergibt dann:

4	D
4	13
8 4 2 1	8 4 2 1
0 1 0 0	1 1 0 1

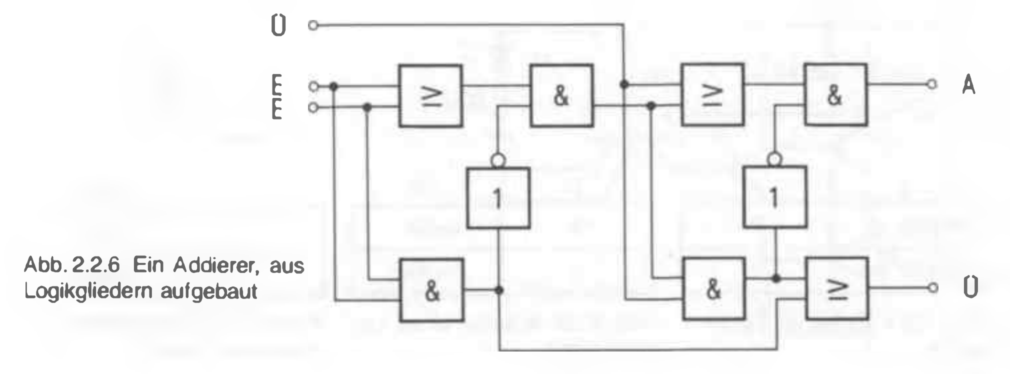


Abb. 2.2.6 Ein Addierer, aus Logikgliedern aufgebaut

Übrigens gibt es Taschenrechner, die Umwandlungen in verschiedene Zahlensysteme und auch Logikbefehle ausführen können.

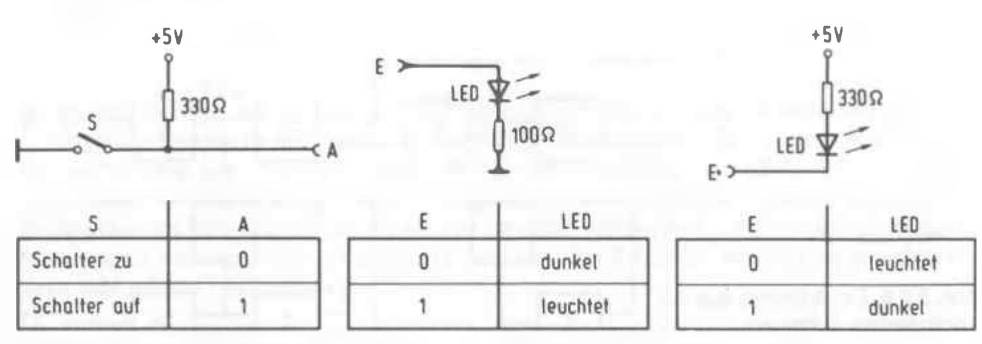
2.3 Definition der Signalpegel

Bei unseren Schaltungen arbeiten wir im allgemeinen mit TTL-Signalen. Das bedeutet 0 V für eine logische 0 und 5 V für eine logische 1. Die meisten TTL-Gatter liefern aber nicht genau diese Werte, sondern einen Pegel von 2,4 bis 5 V für eine logische 1 (was auch als High-Pegel bezeichnet wird) und 0 bis 0,7 V für eine logische 0 (Low-Pegel genannt). Die Eingänge der TTL-Gatter akzeptieren ebenfalls diesen Bereich, so daß die Signalpegel auch wieder richtig verstanden werden.

Bei manchen integrierten Schaltungen kann es aber sein, daß sie einen höheren Pegel als 2,9 V für ein 1-Signal benötigen. Dann findet man bei den Eingängen meist einen sogenannten PULL-UP-Widerstand nach + 5 V. Er hat die Aufgabe, den Pegel etwas nach 5 V zu „ziehen“. Der Widerstandswert liegt i. a. zwischen 1 kΩ und 330 Ω, je nach Art der Eingangsstufe. Warum ist dies überhaupt möglich? Die TTL-Ausgänge besitzen in der 5 V-Leitung der Ausgangsstufe einen internen Widerstand, hingegen bei der 0 V-Zuführung der Ausgangsstufe nicht. Wird extern ein Pull-Up-Widerstand angebaut, so steigt die Spannung bei einem 1-Signal an, bleibt bei einem 0-Signal aber fast unverändert auf dem Wert, den sie ohne Pull-Up-Widerstand hätte.

Zur Ansteuerung von TTL-Gattern ist in *Abb. 2.3.1* eine kleine Schaltung gezeigt, die die richtigen Signalpegel erzeugt. Bei geschlossenem Schalter liegt ein 0-Signal am Ausgang; bei geöffnetem ein 1-Signal. Ein solcher Schalter hat allerdings den Nachteil, daß er beim Schließen prellt: Beim Schließvorgang gibt er mehrere Male Kontakt und erzeugt damit am Ausgang eine Pulsfolge mit einer Dauer von bis zu 10 ms (je nach Schalter). Für manche TTL-Schaltungen, wie z. B. Zähler, ist sie daher nicht zu gebrauchen; wir werden später noch eine bessere Schaltung kennenlernen.

Um Signal-Pegel optisch darstellen zu können, benötigen wir auch eine Ausgabeeinheit. *Abb. 2.3.2* zeigt eine solche Schaltung mit einer Leuchtdiode. Liegt der Eingang auf einem 0-Signal, so ist die LED dunkel; bei einem 1-Signal leuchtet sie. Einen Nachteil besitzt diese Schaltung noch: wir wissen bereits, daß TTL-Schaltungen bei einem 1-Signal einen Widerstand in der Ausgangsleitung haben. Dies führt dazu, daß die LED nicht sehr hell leuchtet. Daher ist in *Abb. 2.3.3* eine andere Schaltung gezeigt. Diesmal wird die LED immer dann leuchten, wenn ein



S	A
Schalter zu	0
Schalter auf	1

E	LED
0	dunkel
1	leuchtet

E	LED
0	leuchtet
1	dunkel

Abb. 2.3.1 Einfacher Signalgeber

Abb. 2.3.2 Anzeige eines Logikzustands

Abb. 2.3.3 Anzeige eines Logikzustands invers

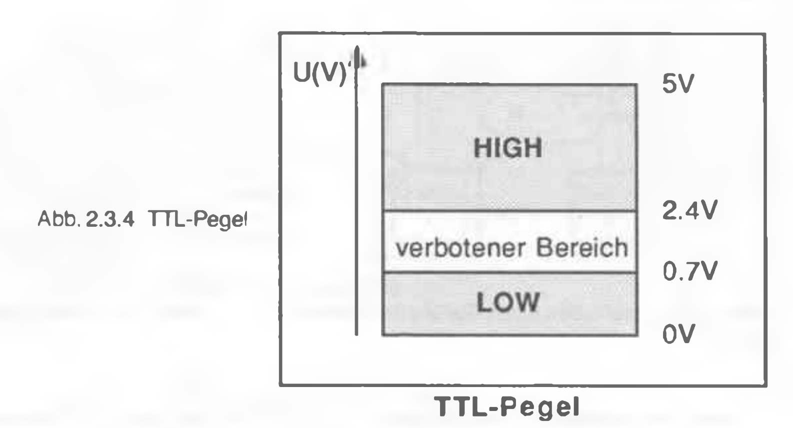


Abb. 2.3.4 TTL-Pegel

0-Signal am Eingang anliegt; sie ist dunkel bei einem 1-Signal. Doch halt, nun ist die Logik umgedreht. Dafür leuchtet die LED auch mit dem größeren Widerstand heller als im vorigen Beispiel. Diese Schaltung wird i. a. bei Mikrorechnerausgaben für die Ansteuerung von LEDs verwendet. Der Nachteil, daß die Anzeige verkehrt (invers) zu der vorherigen ist, spielt dabei keine Rolle, da sich dies per Programm leicht ausgleichen läßt, wie wir später noch sehen werden.

Abb. 2.3.4 zeigt eine Grafik zu den Spannungsbereichen beim TTL-Pegel.

2.4 Bus-Schaltkreise

Bisher wird jeder, der Digitaltechnik schon kannte, mühelos gefolgt sein. Nun kommen wir zu speziellen Bausteinen, die besonders für Mikrorechner-Schaltungen verwendet werden. Dort genügt eine reine Logik-Funktion nicht, sondern es gibt auch Funktionen, die darüber hinausgehen. So ist das mit den sogenannten Bus-Treibern. Mit Standard-Logikschaltungen können wir beliebige Schaltungsnetze aufbauen. Bei Mikrorechnern gibt es auch noch eine andere Struktur, die sogenannte Bus-Struktur. Dazu betrachten wir zunächst die Ausgangsstufe eines Standard-Bausteins. *Abb. 2.4.1* zeigt das Innenleben eines Nand-Gatters 7400. Bei der Ausgangsstufe ist immer einer der beiden Transistoren T3 oder T4 leitend. Damit liegt am Ausgang der Signalpegel 1 oder 0 an.

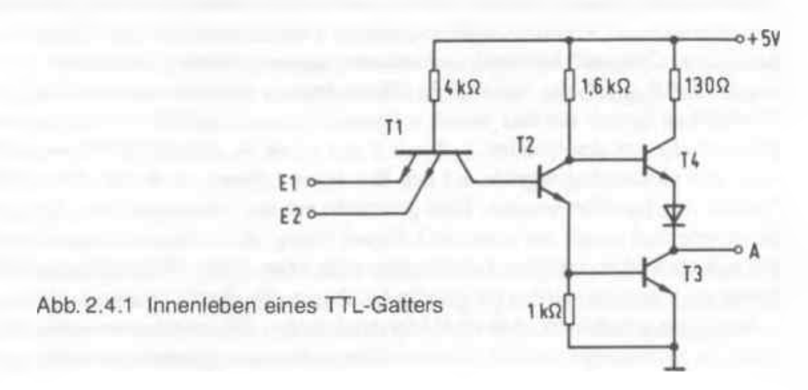


Abb. 2.4.1 Innenleben eines TTL-Gatters

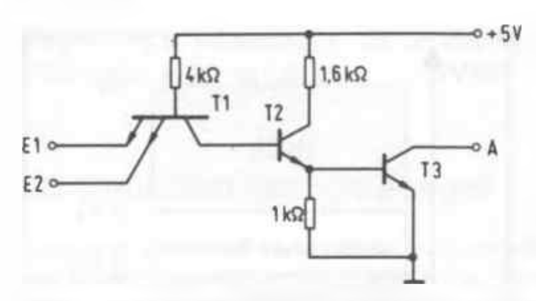


Abb. 2.4.2 Innenleben eines Gatters mit Open-Collector

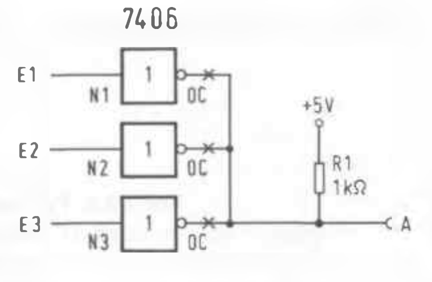


Abb. 2.4.3 Wired-Or-Schaltung

Wird der Transistor T4 weggelassen, so ergibt sich ein Open-Collector-Ausgang. *Abb. 2.4.2* zeigt die Innenschaltung des ICs 7401, das vier solche Open-Collector-Nand-Gatter beinhaltet. Wenn an den beiden Eingängen El und E2 gleichzeitig ein 1-Signal anliegt, so ist die Nand-Funktion erfüllt, und am Ausgang liegt ein 0-Signal. Liegt aber eine andere Eingangsbelegung vor, so leitet T2 nicht, und der Ausgang ist offen. Um damit weiterarbeiten zu können, wird ein Pull-Up-Widerstand benötigt, der auch bei offenem Zustand den Signal-Pegel 1 garantiert. Wozu soll das nun gut sein? Mit diesen Open-Collector-Schaltungen lassen sich neue Schaltungsstrukturen bilden.

Dazu betrachten wir *Abb. 2.4.3,* dort sind drei Ausgänge der Open-Collector-Inverter 7406 zusammengeschaltet. Mit normalen Gattern geht das nicht, da sich zwischen zwei Ausgängen ein Kurzschluß bilden würde. Wir wollen versuchen, die Logikfunktion dazu aufzustellen. Liegt El auf einem 1-Pegel, so ist der Ausgang des Gatters Nl auf 0. Wegen der Open-Collector-Eigenschaften gilt das auch für den Ausgang A der gesamten Schaltung. Damit gilt: Falls einer der Eingänge auf einem 1-Signal liegt, so liegt der Ausgang A auf einem 0-Signal. Ein 1-Signal am Ausgang ist nur dann vorhanden, wenn an allen Eingängen ein 0-Signal liegt, dann wird der Pull-Up-Widerstand Rl am Ausgang das 1-Signal erzeugen. Übrigens wird durch das X am Ausgang gezeigt, daß es sich um eine Open-Collector-Schaltung handelt. In der einschlägigen Literatur ist das leider nicht einheitlich. In diesem Buch wird nur dieses Zeichen verwendet.

Die Schaltung liefert also die NOR-Funktion. Der Vorteil einer solchen Schaltungsart liegt darin, daß die einzelnen Gatter N1, N2 oder N3 auch räumlich voneinander getrennt liegen können und nur durch die Ausgangsleitung miteinander verbunden sind. Bei Mikrorechnern gibt es die Aufgabe, Information (in Form von 0- und 1-Signalen) über Leitungen von einem Teil der Schaltung zur anderen und zurück zu übertragen, wobei die Information von verschiedenen Untereinheiten kommen kann. Für diese Aufgabe eignet sich eine Bus-Struktur. *Abb. 2.4.4* zeigt ein Blockschema. Von den Quellen 1, 2 und 3 sollen Daten zum Ziel übertragen werden. Wie läßt sich dies realisieren? Natürlich mit unseren Open-Collector-Bausteinen.

Abb. 2.4.5 zeigt die Schaltung. Dabei besteht der Bus hier nur aus einer Leitung. In Wirklichkeit besteht ein Bus jedoch normalerweise aus mehreren Leitungen. Beliebige Logikpegel kommen von den Quellen 1, 2 und 3 und sollen an das Ziel durchgeschaltet werden. Damit nicht alle gleichzeitig Signale auf den Bus legen können, muß eine Freigabe für die einzelnen Quellen durchgeführt werden. Dies geschieht mit den Eingängen „frei" 1, 2 und 3. Von diesen Eingängen darf immer nur einer ein 1-Signal führen, die anderen müssen ein 0-Signal haben. Die Auswahl geschieht mit einer Adreßlogik, zu der aber weitere Bus-Leitungen benötigt werden, wie später noch gezeigt wird. Das Prinzip ist aber durch diese Schaltung gezeigt.

Noch sehr unschön ist, daß ein Widerstand nach + 5 V geschaltet werden muß; besser wäre es, wenn die Signalpegel von den Gattern schon richtig erzeugt würden. Auch dies ist möglich, dazu

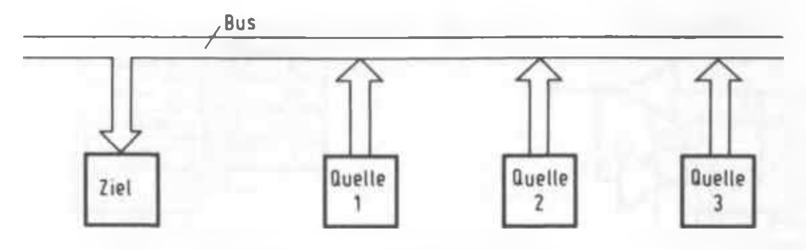


Abb. 2.4.4 Aufbau eines Bus-Systems

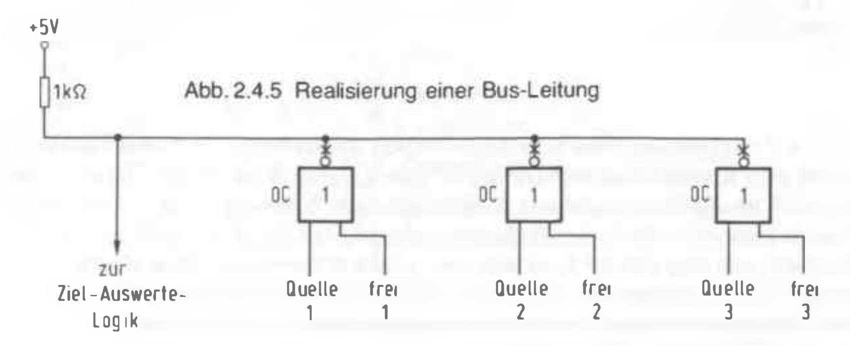


Abb. 2.4.5 Realisierung einer Bus-Leitung

74 LS 244 / 74 LS 367

E A Fr

0	0	0
1	0	1
0	1	offen
1	1	offen

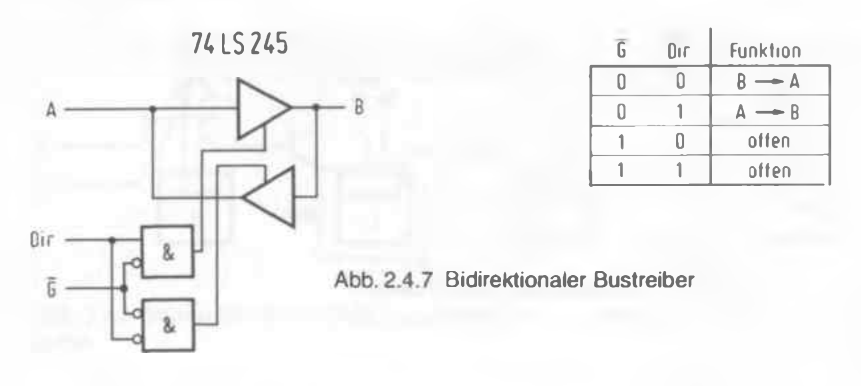
Bus-Treiber

Abb. 2.4.6 Tristate-Treiber

gibt es die sogenannten TRI-State-Ausgänge. In der Ausgangsstufe sind dann wieder beide Transistoren vorhanden, doch ist es mit einem zusätzlichen Eingang möglich, beide Transistoren stromlos zu schalten und dabei wieder den gewünschten offenen Zustand zu erreichen. *Abb. 2.4.6* zeigt die Funktionstabelle und das Schaltzeichen des Bus-Treibers 74367. Liegt der Freigabe-Eingang auf 0, so wird der Signalpegel vom Eingang E auf den Ausgang A durchgeschaltet. Liegt der Freigabe-Eingang Fr auf 1, so ist der Ausgang im offenen Zustand unabhängig vom Eingang E.

Bi-direktionale Bustreiber

Nun kann man sich leicht vorstellen, daß zwei solche Bustreiber auch gegeneinander geschaltet werden können, also der Ausgang des einen Gatters mit dem Eingang des anderen und umgekehrt. Werden die Freigabeeingänge abwechselnd geschaltet, so wird einmal von der einen Seite zur anderen und im anderen Fall umgekehrt übertragen.



$\overline{G}$	Dir	Funktion
0	0	B → A
0	1	A → B
1	0	offen
1	1	offen

Abb. 2.4.7 Bidirektionaler Bustreiber

Abb. 2.4.7 zeigt einen solchen Schaltkreis (74245). Dieser besitzt die beiden Ein-/Ausgänge A und B und zwei Kontroll-Eingänge Dir und G-Quer. Liegt an G-Quer ein 1-Signal, so sind beide Treiberstufen im offenen Zustand und damit die gesamte Schaltung. Liegt G-Quer auf einem 0-Pegel, so ist immer eine der beiden Treiberstufen freigegeben. Liegt Dir auf 0, so wird von B nach A übertragen; und liegt Dir auf 1, so wird von A nach B übertragen. Diese Bus-Schaltung wird verwendet, um an einen Datenbus Signale in beiden Übertragungsrichtungen anlegen zu können.

In dem Schaltkreis 74LS245 sind acht bi-direktionale Bustreiber und ein gemeinsamer Dir- und G-Quer-Anschluß untergebracht. Damit kann auf einen 8 Bit breiten Datenbus zugegriffen werden. Übrigens kann auch der 74LS645 eingesetzt werden, der lediglich einen höheren Ausgangsstrom liefern kann.

2.5 Flip-Flop-Schaltungen

Die bisherigen Gatterschaltungen waren nicht in der Lage, Signalzustände zu speichern. Dies soll sich nun ändern. Es sollen hier aber nur einige von vielen Möglichkeiten dargestellt werden.

D-Flip-Flop

Abb. 2.5.1 zeigt das Schaltbild und die Wahrheitstabelle eines D-Flip-Flops. Es hat folgende Eigenschaft: der Signalpegel, der während der ansteigenden Signalflanke des Taktsignals an dem D-Eingang anlag, wird in das Flip-Flop übernommen. Als ansteigende Signalflanke wird dabei ein Wechsel des Pegels von 0 auf 1 bezeichnet. Man nennt den Übergang auch positive Signalflanke. Das Flip-Flop besitzt noch zwei zusätzliche Eingänge, einen Setz- und einen Rücksetz-Eingang. Damit ist es möglich, durch einen kurzen Puls auf 0, das Flip-Flop in eine bestimmte Lage zu bringen. Ein Puls auf 0 am PRESET-Eingang setzt den Ausgang 0 auf 1 und Q-Quer auf 0. Ein Puls auf 0 am Eingang CLEAR bewirkt das Gegenteil. Die beiden Eingänge liegen während des Takt-Betriebes auf 1. Die Besonderheit eines solchen Flip-Flops liegt darin, nur auf den Übergang am Takt zu reagieren, nicht etwa auf einen statischen Pegel. Dann erst ist es möglich, solche Elemente zu verketten. *Abb. 2.5.2* zeigt ein Beispiel. Der Ausgang des Flip-Flops 1 ist mit dem D-Eingang des nachfolgenden Flip-Flops verbunden. Dadurch ergibt sich ein Schieberegister. Dies hat folgende Eigenschaften: Ein Signalpegel am Eingang des Schieberegi-

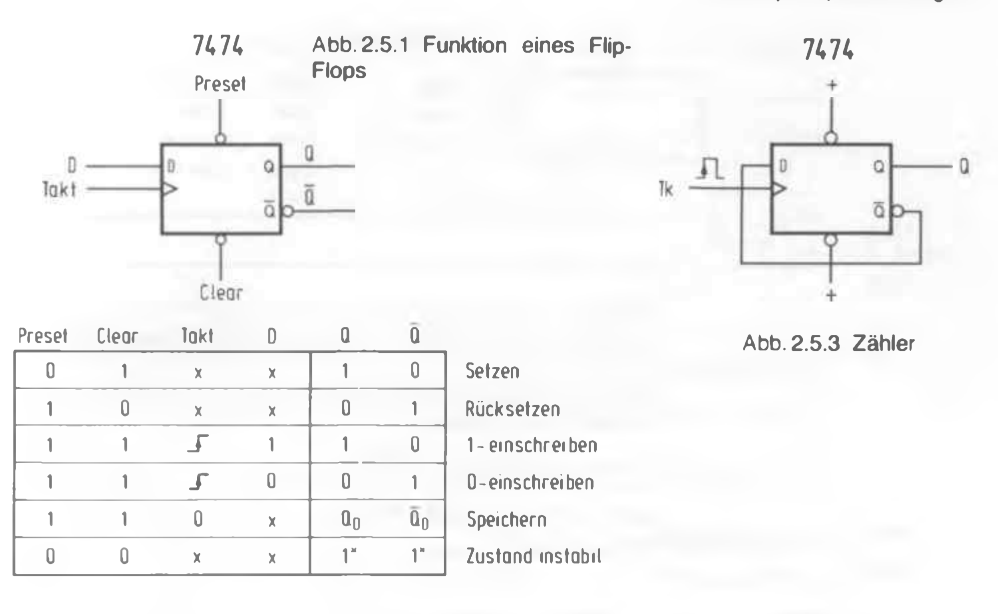


Abb. 2.5.1 Funktion eines Flip-Flops

Preset	Clear	Takt	D	Q	Q̄	
0	1	x	x	1	0	Setzen
1	0	x	x	0	1	Rücksetzen
1	1	↑	1	1	0	1-einschreiben
1	1	↑	0	0	1	0-einschreiben
1	1	0	x	Q_0	$\bar{Q}_0$	Speichern
0	0	x	x	1*	1*	Zustand instabil

Abb. 2.5.3 Zähler

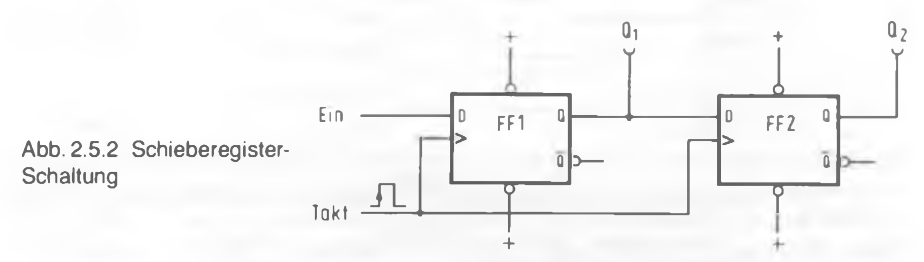


Abb. 2.5.2 Schieberegister-Schaltung

sters wird bei jeder ansteigenden Taktflanke von einem zum nächsten Flip-Flop verschoben. Liegt beispielsweise am Eingang der logische Pegel 1, so erscheint dieser Pegel nach dem ersten Taktpuls am Ausgang Ql und nach dem zweiten Taktpuls am Ausgang Q2. Entsprechend kann der Eingangspegel nach jeder Übernahme wechseln; er wird dann wieder bei der nächsten positiven Signalflanke nach rechts verschoben. An Ql und Q2 liegen dann die Pegel, die zuvor einmal nacheinander am Eingang gelegen haben, parallel an. Damit wurde eine sogenannte Serien-Parallel-Wandlung erreicht. Diese spielt bei den Mikrorechnern eine große Rolle, wie wir später noch sehen werden.

Eine andere Grundschaltung läßt sich bei der Verwendung eines D-Flip-Flops ebenfalls erreichen. *Abb. 2.5.3* zeigt das Schaltbild. Diesmal wird der Ausgang Q-Quer an den Eingang D desselben Flip-Flops geschaltet. Es wird dadurch erreicht, daß sich bei jeder positiven Flanke an dem Takt-Eingang der Zustand des Ausgangssignals ändert. War es zuvor auf 1, so wird es danach auf 0 liegen und umgekehrt. Die Schaltung wirkt als Zähler. Sie ist ebenfalls ein Frequenzteiler. Liegt am Takt-Eingang eine Frequenz von 4 MHz, so wird am Ausgang eine Frequenz von 2 MHz erscheinen.

Solche Zähler gibt es aber auch schon fertig integriert mit mehreren Stufen in einem Gehäuse. *Abb. 2.5.4* zeigt ein Beispiel. Der Schaltkreis 74161 enthält vier Zählerstufen, die ähnlich wie im vorherigen Beispiel aufgebaut sind, jedoch eine Reihe weiterer Eigenschaften besitzen. Die Zählerstufe kann von 0 bis 15 zählen, ferner besitzt sie vier Eingänge, über die sich der

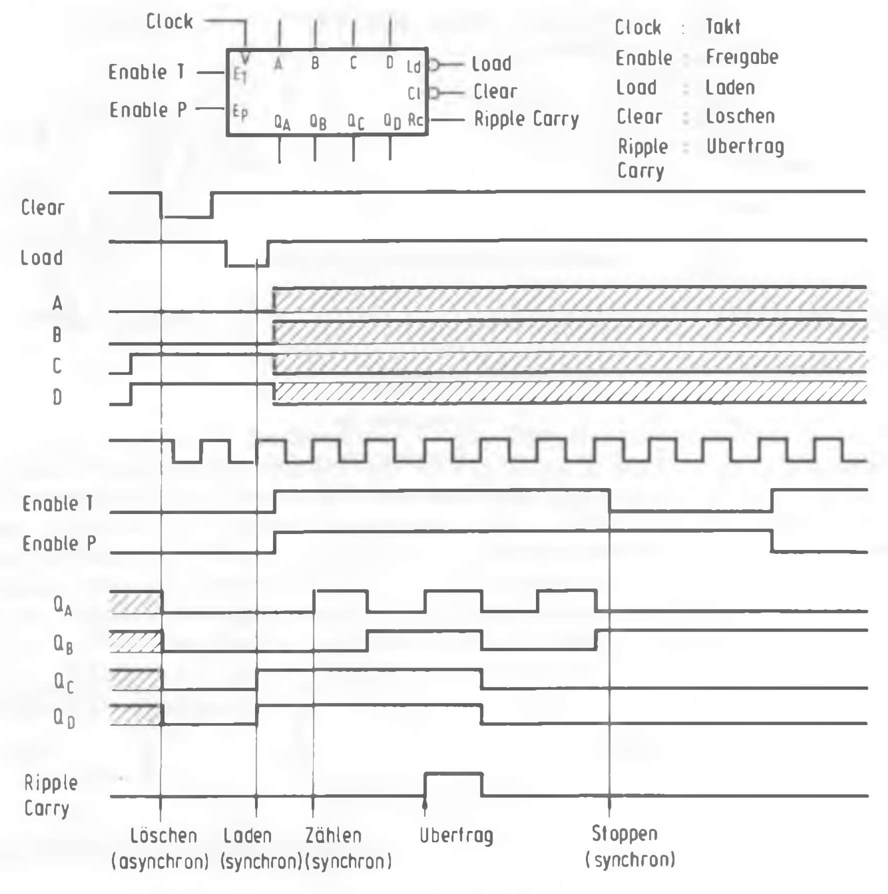


Abb. 2.5.4 Ein integrierter Vierfach-Zähler

Zählerstand vorbelegen läßt. Dazu wird ein Puls auf 0 an den Load-Eingang gelegt. Mit dem Eingang Clock kann der Zählerstand pro positive Signalflanke um eins erhöht werden. Dazu müssen die Freigabe-Eingänge „Enable T" und „Enable P" beide auf 1 liegen. Die Eingänge dienen dazu, mehrere Zählerschaltungen hintereinander anzuordnen. Der Ausgang „Ripple Carry" liefert ein Übertragungssignal, sobald der Zählerstand 15 erreicht ist und beim nächsten Takt der Zähler wieder den Stand 0 annehmen wird. Mit dem Eingang Clear kann der Zähler auf 0 gesetzt werden. Dies ist wichtig, um z. B. nach dem Stromeinschalten einen definierten Anfangszustand zu erreichen. Es könnte auch durch Laden einer 0 mit Hilfe von Load erreicht werden. Um diesen Baustein besser kennenzulernen, werden Experimente damit sehr empfohlen.

RS-Flip-Flop

Eine wesentlich einfachere Flip-Flop-Schaltung zeigt *Abb. 2.5.5*. Es handelt sich um ein RS-Flip-Flop, eine Abkürzung von Rücksetz-Setz-Flip-Flop. Mit einem solchen Flip-Flop kann zum

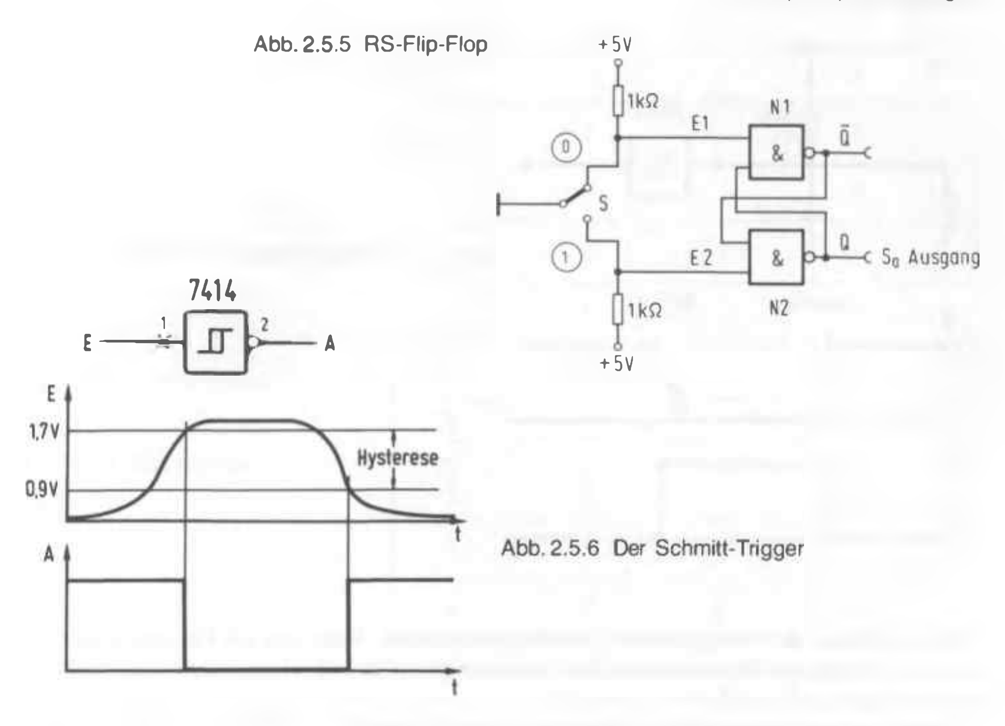


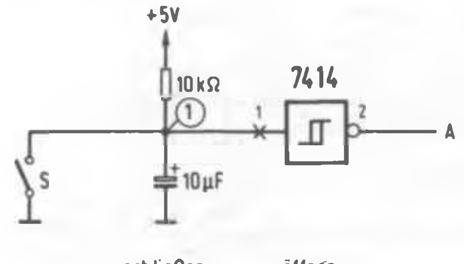
Abb. 2.5.5 RS-Flip-Flop

Abb. 2.5.6 Der Schmitt-Trigger

Beispiel ein Schalter entprellt werden: Wir denken uns zunächst einmal den Schalter weg. Dann soll z. B. der Ausgang Q auf 0 liegen. Damit wird Q-Quer auf 1 liegen und deshalb Q wieder auf 0. Umgekehrt fangen wir mit Q-Quer auf 0 an, so muß Q auf 1 liegen und deshalb Q-Quer auf 0, da die beiden anderen Eingänge des Flip-Flops auf 1 liegen. Das Flip-Flop hat also zwei stabile Zustände. Wird nun einer der beiden zusätzlichen Eingänge auf 0 gelegt, so kann das Flip-Flop in den anderen Zustand kippen. Liegt Q auf dem Wert 0 und wird 0 an den Eingang E2 gelegt, so liegt anschließend der Wert 1 an diesem Ausgang. Mit E1 kann durch einen Puls auf 0 umgekehrt wieder 0 am Ausgang Q geschaltet werden. Der Schalter S tut nun genau das. Es wird abwechselnd E1 oder E2 auf 0 gelegt und damit folgt der Ausgang Q genau der Schalterstellung (Q-Quer folgt invers dazu). Das Prellen des Schalters wird unterdrückt, da der Schalter, ein Umschalter, eigentlich drei Signalzustände besitzt. Einmal Kontakt oben, kein Kontakt und Kontakt unten. Gibt er nun mehrere Male Kontakt, so kann er dies nur zwischen zwei von diesen Zuständen tun – als Prellen zwischen oberem Kontakt und dem offenen Zustand und Prellen zwischen dem unteren Kontakt und offen. Das Flip-Flop wechselt den Zustand aber nur bei einem kompletten Wechsel des Schalters von dem oberen auf den unteren Kontakt oder umgekehrt. Dadurch wird das Prellen vollständig unterdrückt.

Schmitt-Trigger

Ein Baustein, der in der Digitaltechnik auch eine große Bedeutung hat, ist der Schmitt-Trigger. Er kann aus einem analogen Signal ein digitales machen. *Abb. 2.5.6* zeigt das Schaltbild und ein Impulsdiagramm. Eine am Eingang langsam ansteigende Spannung wirkt erst beim Erreichen einer bestimmten Schaltschwelle auf den Ausgang. Abgebildet ist ein invertierender Schmitt-



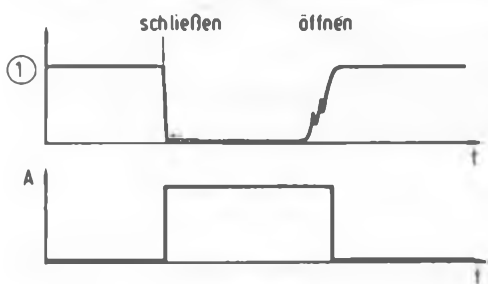


Abb. 2.5.7 Entprellen eines Schalters

Trigger, bei dem der Ausgang dann ein 0-Signal annimmt. Sinkt nun die Eingangsspannung wieder ab, so reagiert der Ausgang erst bei einer niedrigeren Schwelle als beim Ansteigen, und das Ausgangssignal geht wieder auf 1 zurück.

Durch diese unterschiedlichen Schwellen, deren Abstand auch Hysterese genannt wird, können Störungen, die sich am Eingang befinden und die Hysteresespannung nicht überschreiten, unterdrückt werden.

Daher eignet sich auch der Schmitt-Trigger zum Entprellen eines Schalters.

Abb. 2.5.7 zeigt die Realisierung. Ein Schalter, der hier nur noch einen Schließ-Kontakt zu haben braucht, ist mit einer Kondensator-Widerstandskombination verbunden. Wird der Schalter geschlossen, so wird der Kondensator schnell entladen, und am Eingang des Schmitt-Triggers liegt ein 0-Signal. Nun kann sich aber der Kondensator auch beim Prellen des Schalters über den Widerstand nicht mehr so schnell aufladen und daher wird der Einschaltvorgang entprellt. Beim Loslassen des Schalters prellt dieser zunächst und entlädt den Kondensator mehrere Male nacheinander. Erst wenn das Prellen aufgehört hat, lädt sich der Kondensator über die Schwellenspannung des Schmitt-Triggers auf.

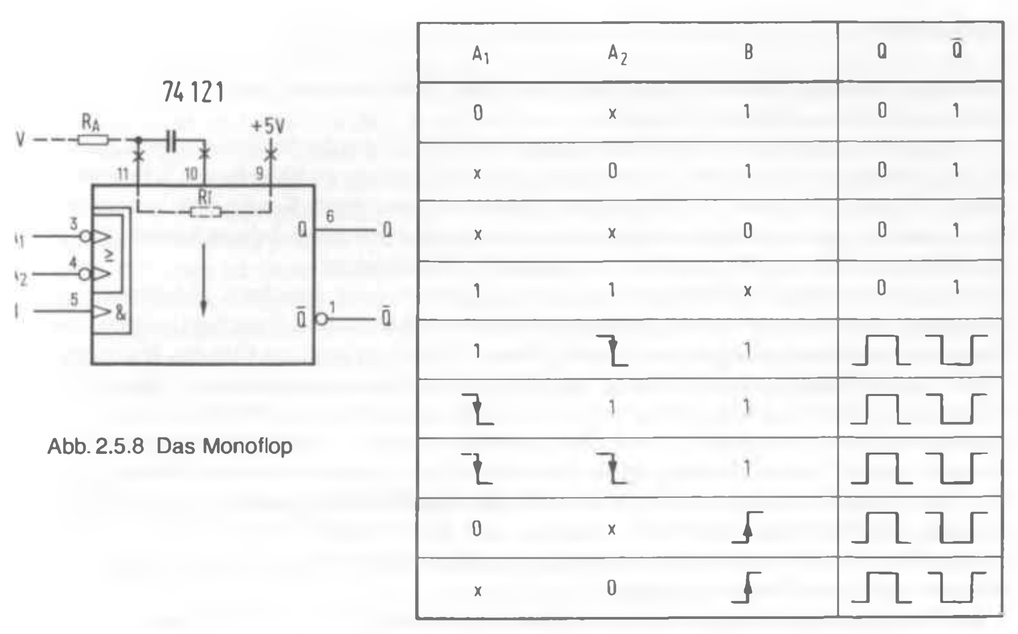
Am Ausgang des Schmitt-Triggers liegt ein entprelltes Schaltersignal an.

Die Schaltung arbeitet natürlich nur dann einwandfrei, wenn Kondensator und Widerstand entsprechende Dimensionierung aufweisen. Ist der Kondensator zu klein, so kann er sich möglicherweise zwischen den Prellpausen wieder voll aufladen und verursacht Störungen. Ist er zu groß, kann der Schalter nicht unmittelbar nacheinander betätigt werden, da das Loslassen des Schalters wie eine lange Prellpause wirkt.

Die im Schaltbild angegebene Dimensionierung der Werte ist im allgemeinen jedoch ausreichend.

Monoflop

Genauso wichtig wie der Schmitt-Trigger ist das Monoflop. Es reagiert auf einen Signalwechsel. Dann wird ein Impuls ausgelöst. *Abb. 2.5.8* zeigt das Schaltbild eines integrierten Bausteins



A_1	A_2	B	Q	$\bar{Q}$
0	x	1	0	1
x	0	1	0	1
x	x	0	0	1
1	1	x	0	1
1	↓	1	⊓	⊔
↓	1	1	⊓	⊔
↓	↓	1	⊓	⊔
0	x	↑	⊓	⊔
x	0	↑	⊓	⊔

x = Pegel spielt keine Rolle, kann 0 oder 1-Signal sein

Abb. 2.5.8 Das Monoflop

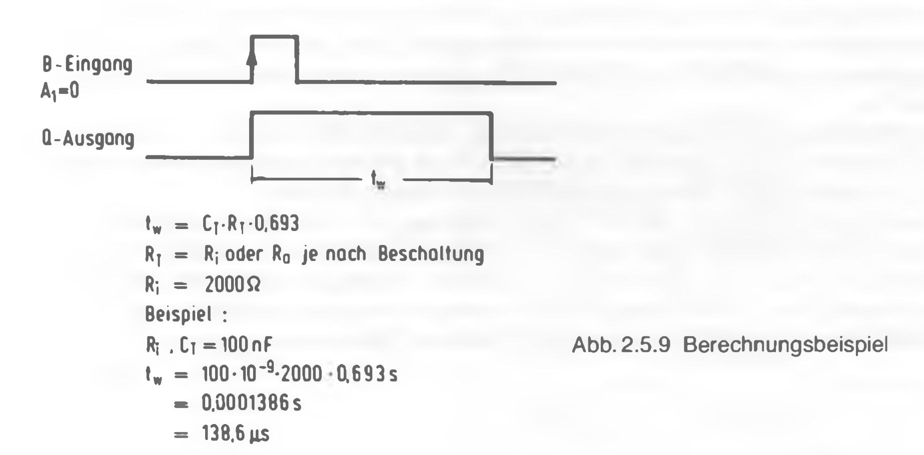


Abb. 2.5.9 Berechnungsbeispiel

74121. Der Baustein besitzt drei Eingänge. Zwei davon reagieren auf eine negative Flanke (Signalwechsel von 1 auf 0), einer auf einem positiven Signalwechsel. Normalerweise wird nur immer einer von beiden Eingängen verwendet.

Abb. 2.5.9 zeigt, was bei einem Signalwechsel passiert. Am Eingang B wird ein Signal von 0 auf 1 wechseln. Dann wird das Monoflop ausgelöst, wenn A1 und A2 auf 0 lagen. Am Ausgang Q erscheint ein positiver Impuls mit einer festen Zeitdauer. Am Ausgang Q-Quer erscheint der gleiche Impuls, jedoch invertiert.

Die Zeitdauer wird durch einen Kondensator und einen Widerstand bestimmt. Der Kondensator wird extern angeschaltet. Ein 2-kΩ-Widerstand ist bereits in das Monoflop eingebaut und kann mit verwendet werden. In der *Abb. 2.5.9* ist gezeigt, wie man die Zeitdauer ausrechnen kann.

Taktoszillator

Mit einer einfachen Gatterschaltung kann man einen Takt erzeugen. *Abb. 2.5.10* zeigt die Schaltung und das abkürzende Symbol.

Als Oszillator werden zwei TTL-Schaltkreise verwendet. Damit der Oszillator einen konstanten Takt erzeugt, wird ein Quarz verwendet, der eine sehr genaue, gleichbleibende Schwingung steuert. Die beiden Inverter 74LS04 sind mit Widerständen und einem Kondensator verschaltet. Hier sei betont, daß es sich nicht mehr um eine rein logisch arbeitende Schaltung handelt. Durch den Widerstand R1 und R2 entsteht eine sogenannte Rückkopplung.

Betrachten wir ein Gatter einmal isoliert. *Abb. 2.5.11* zeigt einen Ausschnitt. Nehmen wir an, am Eingang des Inverters liege ein 1-Signal nach dem Stromeinschalten. Dann liegt aufgrund der logischen Funktion am Ausgang ein 0-Signal. Dieses 0-Signal gelangt nun über den Widerstand wieder an den Eingang, also wird wieder invertiert und am Ausgang erscheint ein 1-Signal, das wieder an den Eingang gelangt und so weiter. Die Schaltung schwingt also. Mit welcher Frequenz sie schwingt, kann man nicht so genau sagen, das hängt davon ab, wie lange es dauert, bis der Ausgang „merkt", was am Eingang gerade für ein Pegel liegt; man nennt das auch Gatterlaufzeit. Die Gatterlaufzeit ist aber sehr unterschiedlich. Die gesamte Schaltung aus *Abb. 2.5.10* aber ist in der Lage, den Quarz zum Schwingen anzuregen, und dieser bestimmt dann die Frequenz, und zwar dadurch, daß er – vereinfacht ausgedrückt – die Eigenschwingung der Gatterteilschaltungen auf seine bevorzugte Frequenz bremst.

Die Taktschaltung hat aber auch so seine Haken. So arbeitet sie nicht mit allen Quarzfrequenzen gleich gut. Ist die Frequenz zu hoch, so schwingt sie manchmal gar nicht mehr (z. B. 16 MHz); ist sie zu niedrig (kleiner 1 MHz), so schwingt die Schaltung manchmal auf einer Oberwelle, das ist ein ganzzahliges Vielfaches der Quarzgrundfrequenz. Trotz mancher Probleme, die die Schaltung haben kann, wird sie in der Industrie gern eingesetzt, da sie so einfach und preiswert zu realisieren ist.

Es gibt auch komplett integrierte Quarzoszillatoren, die jedoch sehr teuer sind, aber zuverlässig arbeiten. Bei einem 4 MHz Quarz und einem 75LS04 gibt es jedoch keine Probleme und daher werden wir diese Schaltung auch bei unserem Mikrocomputer einsetzen.

Hier noch eine Bemerkung zu der TTL-Bezeichnung LS. Wir haben gelegentlich von unterschiedlichen Gattertypen gesprochen, z. B. 7404 und 74LS04. Es handelt sich um zwei verschiedene Technologien. Die Standard-Serie – ohne LS – benötigt im allgemeinen mehr Strom als die LS-Serie. Die LS-Serie hingegen kann weniger Ausgangsstrom liefern und ist im Durchschnitt etwas langsamer als entsprechende Standard-TTLs.

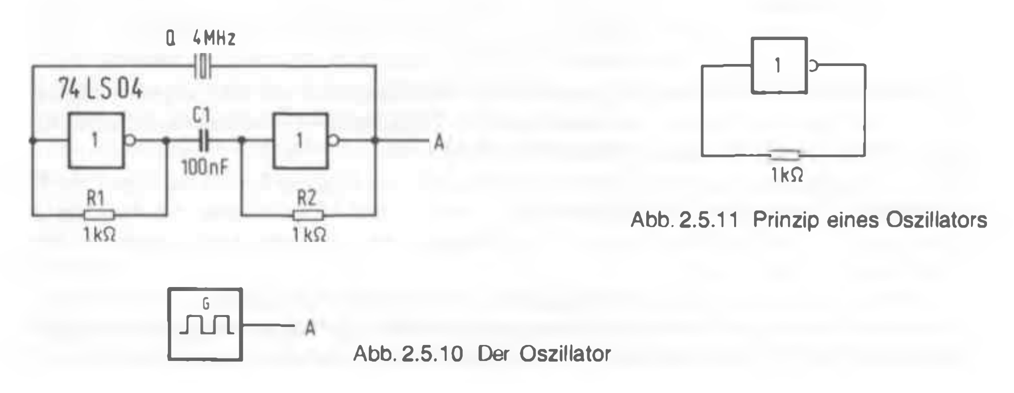


Abb. 2.5.11 Prinzip eines Oszillators

Abb. 2.5.10 Der Oszillator

Im Buch sind die Bezeichnungen immer so angegeben, wie es für die einzelnen Schaltungen vorteilhaft ist. LS-Bausteine sind günstiger, da der Stromverbrauch in den Schaltungen geringer bleibt. Sie haben aber auch einen besonderen Nachteil. Liegt eine leicht negative Spannung am Eingang an, so wird diese wie ein 1-Signal gewertet. Standard-TTLs machen das nicht. Man wird sich nun fragen, woher kommt eine negative Spannung am Eingang? Bei langen Leitungen und weit auseinander liegenden Bauteilen ist dies ganz leicht möglich, denn durch den Spannungsabfall an den Versorgungsleitungen liegt bei einem weit entfernten IC ggf. an seinem Masse-Anschluß eine wenig positive Spannung an. An dem Eingang seiner Logik liegt aber vielleicht genau 0 V, dann ist dieser Eingang negativ bezogen auf die Versorgung – und der LS-Baustein liefert ein falsches Ergebnis. In unseren Mikrorechnerschaltungen ist immer darauf geachtet, daß dies nicht passieren kann. Jedoch sollte man beim Selbstbau – ggf. ohne die Platinvorlagen – darauf achten, die Versorgungsspannung über dicke Drähte oder gar Kupferflächen vorzunehmen, so daß alle ICs bei den Masse-Anschlüssen die gleichen Spannungspegel haben.

2.6 Fragen zur Digitaltechnik

1. Wie kann aus Nand-Gattern ein Oder-Gatter aufgebaut werden?
2. Wie läßt sich mit dem Flip-Flop 7474 ein Schalter entprellen?

Abb. 2.7.1 Schaltzeichen diskreter Elemente

Schaltzeichen	Benennung
—▭—	Widerstand
—‖—	Kondensator
—‖+—	Kondensator mit Polung
—▷|— K	Diode
—▷|—	Leucht - Diode
⊐‖⊏	Transformator
Si —▬—	Sicherung
B C E	Transistor npn
B C E	Transistor pnp

—+—	Kreuzung, unverbunden
—•—	Verbindung

2.7 Schaltzeichen

An dieser Stelle eine kurze Zusammenfassung von verwendeten und gebräuchlichen Schaltzeichen: *Abb. 2.7.1* zeigt die Schaltzeichen von einigen diskreten Bauteilen (wie Widerstand, Transistor), *Abb. 2.7.2* zeigt die Schaltzeichen von Logik-Gattern, dabei sind die alte amerikanische und neue DIN-Norm gegenübergestellt. *Abb. 2.7.3* zeigt Flip-Flop-Schaltsymbole.

Schaltzeichen alt	Schaltzeichen neu	Benennung neu (IC-Beispiel)
	&	UND-Glied (7408)
	≧1	ODER-Glied (7432)
	1	NICHT-Glied (7404)
	&	NAND-Glied (7400)
	≧1	NOR-Glied (7402)
	=1	ANTIVALENZ-Glied (7486)

Abb. 2.7.2 Schaltzeichen von Logik-Elementen

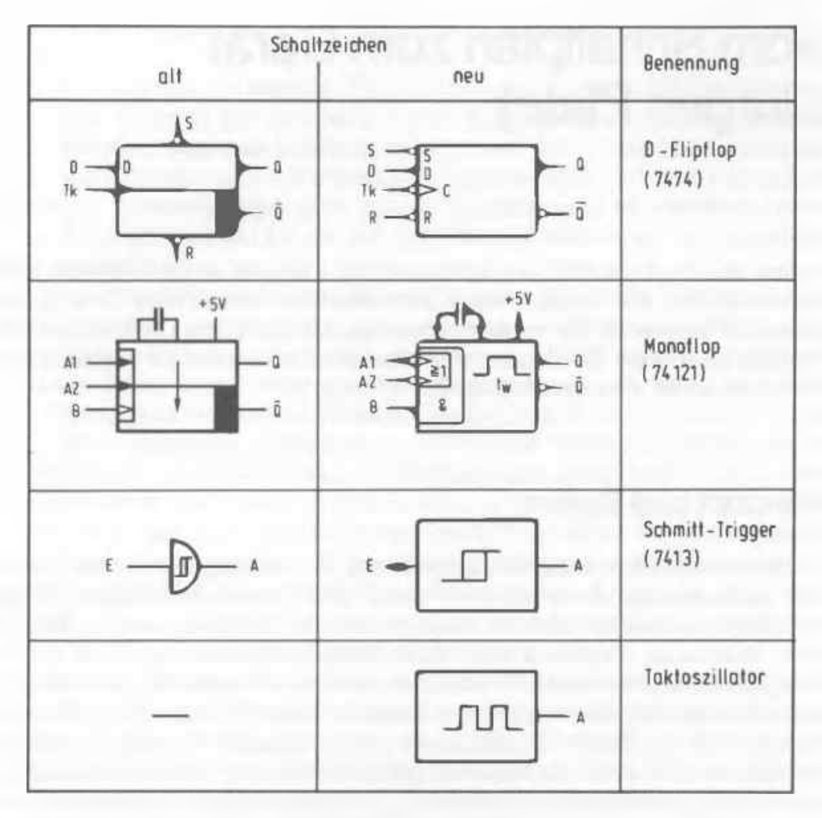


Abb. 2.7.3 Schaltzeichen von Flip-Flop-Schaltungen

3 Vom Schaltplan zum Gerät (Jürgen Plate)

Aller Anfang ist schwer – so heißt das Sprichwort. Ein Computer ist ein komplexes technisches Gerät, dessen Aufbau dem Anfänger einige Schwierigkeiten machen kann. Es ist ja nicht jeder unter Ihnen ein Elektronik-Profi. An einem einfachen Beispiel, einem Logikprüfstift, sollen Sie erste Übungen durchführen. Der Stift selbst ist einfach aufzubauen und das wichtigste Prüfinstrument beim Aufbau der anderen Schaltungen.

3.1 Messen und Bauen

Jedes Gerät ist nach einem Schaltplan aufgebaut. In ihm ist festgelegt, welche Bauelemente verwendet werden und wie sie miteinander verbunden sind. Obwohl die Bauelemente bei gleicher Funktion recht unterschiedlich aussehen können, werden im Schaltplan genormte Schaltsymbole verwendet. So wird ein Widerstand durch ein Rechteck, ein Kondensator durch zwei parallele Striche dargestellt. Manche Bauteile besitzen eine bestimmte Orientierung, sie müssen in einer bestimmten Richtung eingebaut werden. Damit diese Orientierung erkennbar ist, werden sie im Schaltplan markiert: die Elektrolytkondensatoren besitzen eine mit + gekennzeichnete Elektrode, bei den Dioden ist die Katode (die Minusseite) durch die Richtung des Schaltsymbols markiert.

Doch zurück zum Schaltplan, der in *Abb. 3.1.1* wiedergegeben ist. Der Prüfstift besteht aus zwei integrierten Schaltungen, vier Widerständen und vier Leuchtdioden. Die erste integrierte Schaltung enthält vier Inverter, von denen wir nur drei I1, I2, I3 verwenden, die zweite ein Paar Flipflops, von denen nur eins (I1) verwendet wird. Der Schaltplan zeigt also nur die verwendeten Teile der ICs. Die Nummern an den Ein- und Ausgängen der Schaltungsteile der ICs entsprechen den Nummern der Beine des ICs.

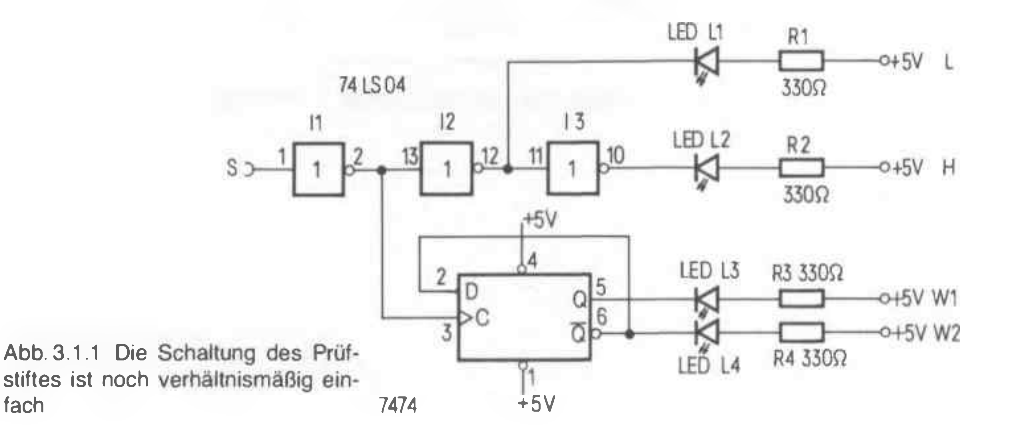


Abb. 3.1.1 Die Schaltung des Prüfstiftes ist noch verhältnismäßig einfach

So funktioniert der Prüfstift

Das Prüfsignal gelangt von der Prüfspitze S an den Inverter I1, der den Rest der Schaltung vom Prüfobjekt trennt. Hinter I1 teilt sich das invertierte Eingangssignal. Im oberen Schaltungsteil wird das Signal mit I2 nochmals invertiert; hinter I2 steht also wieder das Originalsignal zur Verfügung. Von dort geht es zur Leuchtdiode L1 und zum Inverter I3. Ist das Eingangssignal bei S auf 0, leuchtet L1. L2 leuchtet genau dann, wenn das Eingangssignal bei S den Wert 1 besitzt. Die Widerstände R1 und R2 (ebenso R3 und R4) begrenzen den Strom durch die Leuchtdioden auf einen brauchbaren Wert.

Im Prüfstift wird auch ein sogenanntes Flipflop verwendet. Ein solches Schaltglied unterscheidet sich von den vorher schon besprochenen Logikgliedern insofern ganz elementar, als es nicht einfach irgendwelche Eingangssignale verknüpft, etwa wie ein UND aus zwei Signalen ein neues macht, sondern früher einmal vorgekommene Eingangssignale sich merkt und später zum Beispiel in Abhängigkeit davon auf neue Eingangssignale reagiert. Das Flipflop hier besitzt zwei Ausgänge, die so miteinander gekoppelt sind, daß der eine immer den inversen Zustand des anderen einnimmt. Das Flipflop kann nun, so ist es konstruiert, genau zwei Zustände annehmen, die nach außen dadurch sichtbar werden, daß der Ausgang Q einmal 0-Signal führt und einmal 1-Signal. Genau umgekehrt dazu liegen die Signale von $\overline{Q}$. Das Flipflop, das hier verwendet wird, besitzt einen D-Eingang, der mit dem mit dem Pfeil gekennzeichneten Eingang zusammenspielt. Immer dann, wenn das Signal am mit dem Pfeil gekennzeichneten Eingang von 0 auf 1 wechselt, wird das in diesem Augenblick an D anliegende Signal übernommen und intern festgehalten. Q zeigt dann nach außen diesen neuen Zustand, $\overline{Q}$ das Inverse dazu.

Die Rückführung des Signales von $\overline{Q}$ an den D-Eingang ist nun ein raffinierter Kunstgriff. Während der kurzen Anstiegszeit des Signales am sogenannten „Triggereingang" wird zwar das Signal an D ins Innere des Schaltgliedes übernommen. Die Weiterleitung an den Ausgang Q und $\overline{Q}$ geschieht aber mit einer wenn auch sehr kleinen Verzögerung, die gewährleistet, daß der von $\overline{Q}$ angeregte Wechsel des Zustandes, der ja auch $\overline{Q}$ wieder umsteuert, nicht mehr an D registriert wird. Erst bei einem neuen Anstieg der Signalflanke am Triggereingang spielt dieser neue Zustand eine Rolle. Das Fazit: Wenn am Prüfstifteingang S eine Folge von Rechteckimpulsen auftritt, also 01010101010 . . ., dann ändert der Ausgang Q bei jedem Wechsel von 0 auf 1 an S seinen Zustand.

S: 01010101010 . . .
Q: 01100110011 . . . .
$\overline{Q}$: 10011001100

Jeder Wechsel des Eingangssignals von 0 auf 1 läßt das Flipflop kippen. So ergibt sich ein Signalwechsel am Ausgang des Flipflops; wenn zuvor L3 leuchtete, brennt nun L4 und umgekehrt. Während also der obere Teil statische Zustände anzeigt, werden mit dem Flipflop dynamische Vorgänge (Impulse) angezeigt.

Abb. 3.1.2 zeigt die verschiedenen Zustände der LEDs in Abhängigkeit vom Eingangssignal. Bei den LEDs wird in drei Leuchtzustände unterteilt. Zum einen kann die LED hell leuchten, dann dunkel sein und zum dritten leuchtet sie weder hell noch gar nicht, also halbhell.

Liegt ein statisches Eingangssignal vor, so zeigen die LEDs L1 und L2 den Signalpegel an. Leuchtet L1, so liegt ein 0-Signal an, leuchtet dagegen L2, so liegt ein 1-Signal am Eingang (auch wenn dieser offen ist). Die LEDs L3 und L4 spielen dabei keine Rolle. Es leuchtet aber nur eine von beiden LEDs auf. Anders verhält sich das ganze bei einem anliegenden Takt. Ist dieser symmetrisch, so leuchten alle vier LEDs auf. Bei einer Pulsfolge mit 1-Pulsen leuchten L3 und L4, und diesmal ist L2 dunkel (oder fast dunkel). Umgekehrt bei einer 0-Pulsfolge ist L1 fast dunkel.

	L L1	H L2	W1 L3	W2 L4
S = 0-Signal	✱	○	✱	○
S = 0-Signal	✱	○	○	✱
S = 1-Signal	○	✱	✱	○
S = 1-Signal	○	✱	○	✱
S =	⊛	⊛	⊛	⊛
S =	✱	○	⊛	⊛
S =	○	✱	⊛	⊛

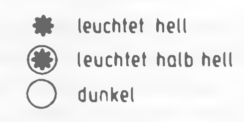


Abb. 3.1.2 Das Schema, nach dem die Leuchtdioden am Prüfstift aufleuchten. Setzen Sie Rot für L, Grün für H ein, Gelb für die Ausgänge am Flip-Flop

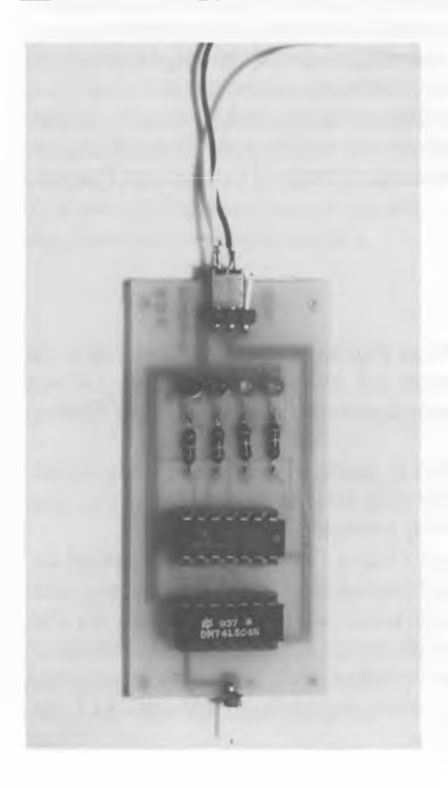
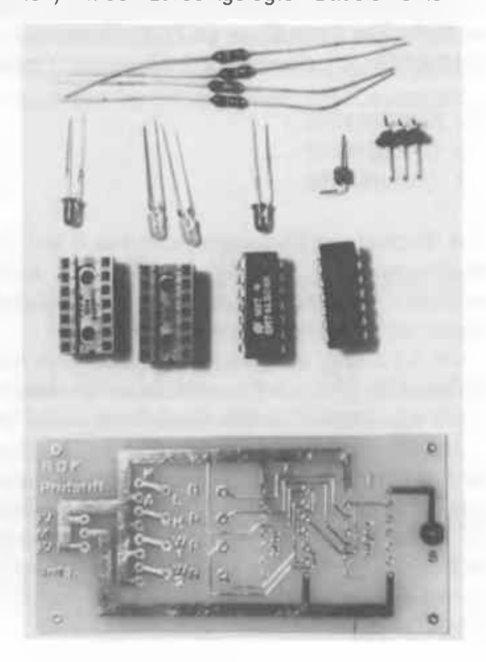


Abb. 3.1.3 Die Bestückungsseite des Teststiftes mit den Bauelementen (oben). Die Lötseite (unten) mit den zurechtgelegten Bauelementen



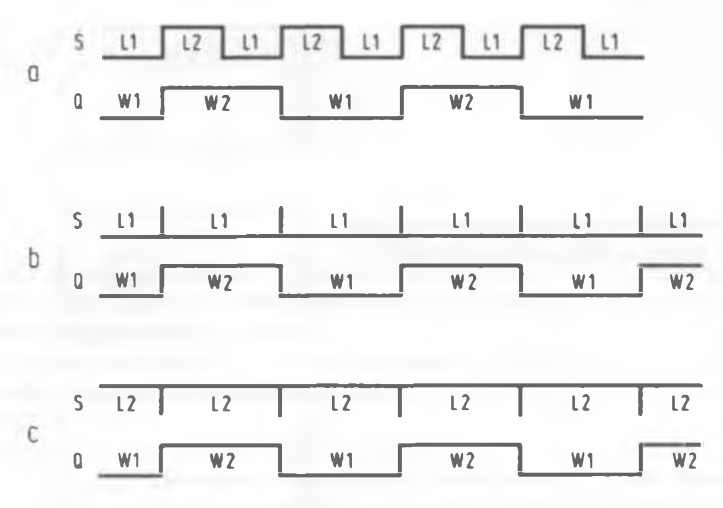


Abb. 3.1.4 Unterschiedliches Verhalten bei Signalen

Abb. 3.1.4 zeigt das Impulsschema zu dieser Schaltung. Durch das Flip-Flop Z1 wird eine unsymmetrische Pulsfolge in eine symmetrische mit halber Frequenz geteilt; daher leuchten L3 und L4 gleich hell.

Der Aufbau ist nicht schwer

Zunächst legen Sie die Teile des Bausatzes und das Werkzeug bereit. Meist brauchen Sie nicht mehr als einen Lötkolben mit gut verzinnter Spitze, Lötzinn, einen kleinen Seitenschneider und eine starke Pinzette. Lassen Sie sich Zeit mit dem Aufbau und arbeiten Sie sorgfältig – dann kann garnichts schiefgehen. Die Platine hat eine Seite, auf der die Bauteile eingesteckt werden und eine Seite, auf der die Bauteileanschlüsse festgelötet werden. Die Lötseite der Platine ist gekennzeichnet, meist mit dem Text „löts.“. Nun werden zunächst die Fassungen für die integrierten Schaltungen eingelötet. Es gibt nun Fassungen, die eine kleine Kerbe an einer der beiden Schmalseiten besitzen. Damit soll gekennzeichnet sein, in welcher Einbaulage das IC auf die Fassung gesteckt werden muß, damit dessen Beine auch mit den richtigen Anschlüssen auf der Platine Kontakt bekommen. Es gibt leider aber auch Sockel, die undeutliche oder gar keine Markierungen tragen. Dann muß man sich vor dem Einbau des Sockels ansehen, wie das IC aufgesteckt werden muß (das ist meist auf der Bestückungsseite der Platine angezeichnet), weil der Sockel oft diese Kennzeichnung verdecken kann. Das IC selbst trägt ebenfalls immer eine Markierung, mit deren Hilfe Pin 1 aufgefunden werden kann. Leider sind diese Markierungen nicht bei allen Fabrikaten gleich.

Abb. 3.1.5 zeigt schematisch ein paar Varianten. Man beginnt immer an der Seite zu zählen, auf der sich die Markierung befindet. Und zwar zählt man gegen den Uhrzeigersinn, wenn das IC von oben betrachtet wird. Zeigt darüber hinaus die Markierung an der Schmalseite auf den Betrachter, dann liegt Pin 1 rechts davon.

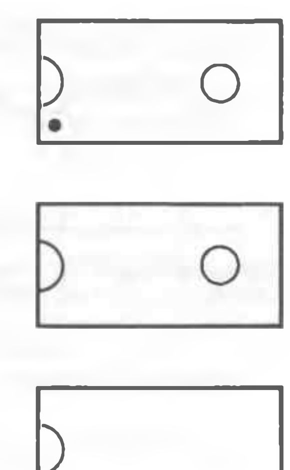


Abb. 3.1.5 Verschiedene Gehäuseformen bei den ICs machen manchmal die Identifikation des Pin 1 schwer

Es wird gelötet

Zunächst löten Sie zwei diagonal gegenüberliegende Beinchen der Fassung an, damit sie nicht herausfallen kann. Danach werden dann alle anderen Beinchen angelötet. Die häufigsten Fehler beim Einlöten von IC-Fassungen sind einerseits Lötbrücken zwischen benachbarten Beinchen oder zwischen Beinchen und einer nahe gelegenen Kontakttierung auf der Platine, andererseits das Vergessen eines der Beinchen – achten Sie darauf. Das beste Prüfinstrument ist hier eine gute Lupe, mit der Sie auch kalte Lötstellen finden können. Anschließend werden die Steckkontakte eingelötet. Nun kommen die Widerstände und Kondensatoren dran. Beim Prüfstift gibt es keine Kondensatoren, trotzdem einige grundsätzliche Anmerkungen darüber. Es gibt ungepolte Kondensatoren, bei denen die Einbaurichtung keine Rolle spielt und Elektrolytkondensatoren, die mit der richtigen Polung eingebaut werden müssen. Die Elektrolytkondensatoren tragen dazu eine Markierung, die entweder auf den Plus- oder den Minuspol hinweist. Sie werden auch zwei Erscheinungsformen kennenlernen, die zylindrischen Becherelkos und die Tantalelkos, die wie Tropfen aussehen. Doch zurück zum Prüfstift. Jetzt werden die Widerstände in die Platine eingesteckt (wie herum ist egal), die Drähte auf der Lötseite etwas abgewinkelt und dann auf ca. 2 mm Länge abgeschnitten. Danach tritt wieder der Lötkolben in Aktion. Was nun noch fehlt, sind die Leuchtdioden. Diese müssen richtig herum eingebaut werden. Je nach Hersteller wird die Katode, die Minusseite, unterschiedlich markiert. Entweder, das Gehäuse ist ein wenig abgeplattet, das Anschlußbein ist länger als das andere oder es ist breiter als das andere. Sie finden die Kathode, die an „Minus" angeschlossen werden muß, sicher heraus, wenn Sie die Anschlüsse innerhalb der Leuchtdiode betrachten. Der „große, flächigaussehende", das ist die Katode *Abb. 1.2.5*.

Tips zur Fehlersuche

Fehler findet man am schnellsten durch systematisches Vorgehen. Beim Prüfstift gibt es beispielsweise zwei relativ unabhängige Einheiten. Ich will die Fehlermöglichkeiten nur in Stichpunkten aufzählen: keine Versorgungsspannung, IC oder Leuchtdiode verkehrt eingebaut, Lötbrücke zwischen zwei Kontakten, kalte Lötstelle, Lötstelle vergessen, Bauteil defekt.

4 Der Mikrorechner

Die Begriffe Mikrocomputer und Mikroprozessor sind heute fast schon in der Umgangssprache enthalten. Was macht eigentlich den Mikroprozessor so bedeutsam? Dazu verfolgen wir einmal kurz die Entstehungsgeschichte.

Ganz am Anfang der Elektronik, hat man Schaltungen mit Röhren zusammengebaut. Etwas später kamen die Transistoren. Diese waren kleiner als Röhren und entwickelten auch nicht mehr so viel Wärme.

Damit konnten die Schaltungen komplexer werden.

Nun kam die Verkleinerungsphase. Transistoren konnte man auf kleine Siliziumplättchen integrieren. Erst einen, dann mehrere. So konnte man schließlich komplette Schaltungen unterbringen. Am Anfang waren das nur einfache Verknüpfungsglieder, dann Flip-Flop-Schaltungen und schließlich Speicher. Die Anzahl der Transistoren, die man unterbringen kann, wächst auch heute noch ständig, man rechnet mit einer Vervierfachung pro Jahr.

Was tut man aber mit so vielen Transistoren? Man kann Spezialschaltungen aufbauen, zum Beispiel Frequenzmesser, Ablaufsteuerungen usw. Nun kam bald ein Problem auf. Die Schaltungen wurden immer spezieller und die Kosten für die Entwicklungen stiegen ständig an, da die Schaltungen immer komplexer wurden.

Spezial-Schaltungen lassen sich aber nicht so oft verkaufen wie Universalschaltungen. Was tun? Da erinnerte man sich der Rechnertechnik. Ein Computer ist doch etwas universelles und kann trotzdem Spezialaufgaben übernehmen, die durch sein Programm bestimmt werden. Der erste Mikrocomputer wurde entwickelt. Er arbeitete noch mit vier Bit und war sehr einfach gehalten. Aber er erfüllte die Erwartungen voll und ganz. Mit zwei Universaleinheiten konnte man nun Spezialaufgaben erfüllen. Die erste Universalschaltung war der Mikroprozessor und die zweite war sein Speicher. In diesen Speicher kann man nun ein Spezialprogramm ablegen, das die ganze Anordnung zur Lösung einer Spezialaufgabe befähigt.

Eine geniale Idee also. Bis heute hat sich an diesem Prinzip nichts geändert. Nun kann man in Ruhe die Integrationsdichte hochtreiben, also komplexere Mikroprozessoren herstellen und komplexere Speicher mit einer größeren Speicherkapazität und das in großen Stückzahlen, denn erst durch das Programm, das unterschiedlich sein kann, wird die Spezialisierung vorgenommen.

Am Anfang wurden die Mikroprozessoren noch zum Steuern und Regeln eingesetzt. Heute verwendet man sie auch in Form von Home- oder Personalcomputern für den Einsatz als Computer, der rechnen kann, mit dem man schreiben kann und der auch zum Spielen gut ist. Der Mikrocomputer konnte so auch den Großcomputer von vielen Plätzen verdrängen.

In diesem Kapitel wollen wir beginnen, unseren eigenen Mikrocomputer aufzubauen. Doch zuvor etwas zur Entstehungsgeschichte dieses Computers. Heute gibt es auf dem Markt eine Vielzahl von Home- und Personalcomputern. Doch diese Vielzahl erinnert mich immer wieder an die Anfänge der integrierten Schaltungstechnik, in der die Vielzahl der Spezialschaltungen Probleme machte. Diese Rechner veralten zudem schnell und sind meist nicht flexibel.

Warum also nicht einen Computer aufbauen, der mitwachsen kann, der ständig erweiterbar bleibt und immer auf dem neuesten Stand der Technik ist.

Dazu braucht man ein modulares System. Das heißt, die Elemente des Computers bestehen aus ganz kleinen Einheiten, die man auswechseln kann.

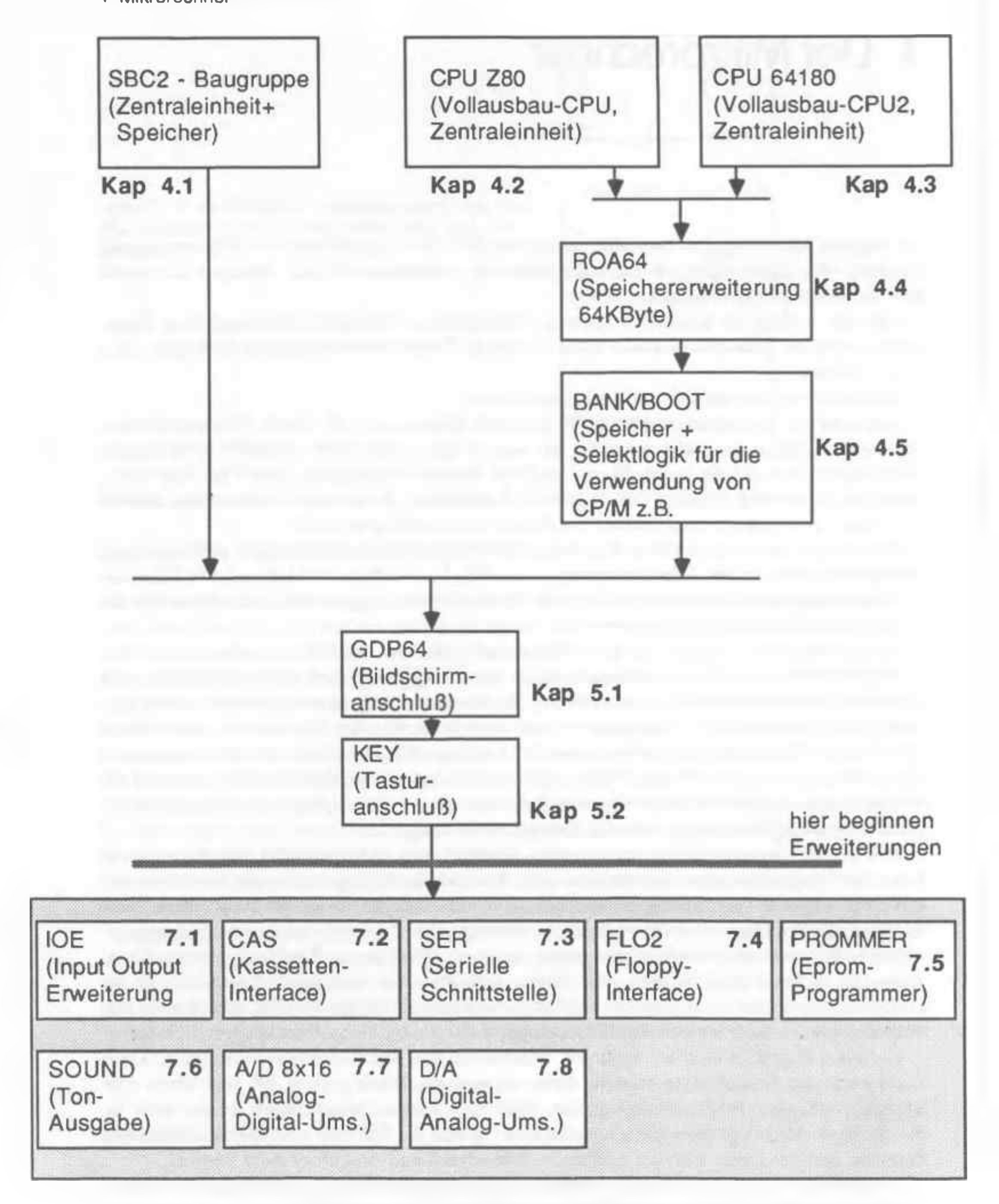


Abb. 4.1 Kapitelübersicht – Aufbau der Baugruppen

Damit wird er aber flexibel. Man kann den Speicher gegen einen größeren austauschen oder sogar die CPU (Zentraleinheit oder Mikroprozessor), wenn einmal eine bessere auf den Markt kommt.

So ein System wollen wir gemeinsam aufbauen und dabei fangen wir mit einem einfachen Mikroprozessor und einer einfachen Schaltung zum Eingewöhnen an. *Abb. 4.1* zeigt in einer Übersicht die Baugruppen, wie sie in diesem Buch vorgestellt werden und wie man sie kombinieren kann.

Die einfachste Baugruppe ist die SBC2, ausgesprochen Single Board Computer Typ 2 oder zu deutsch Computer auf einer Karte.

Die Karte ist ganz einfach gehalten, so daß der Anfänger keine Schwierigkeiten beim Aufbau hat. Wer gleich richtig groß einsteigen will, fängt gleich mit der Vollausbau-CPU an und mit einer Speicherkarte (ROA64). Dieses System kann er dann voll ausbauen, wie schon der Name sagt. Einen dritten Weg gibt es auch noch, wenn man die CPU 64180 verwenden will, so kann man diesen modernen Prozessor einsetzen, der viele Funktionen auf dem Chip integriert hat. Die ersten beiden Karten verwenden den Mikroprozessor vom Typ Z80, so wie er in der Industrie noch immer verwendet wird. Die dritte CPU-Karte verwendet den Baustein 64180, der aber alle Befehle des Z80 enthält und noch ein paar neue dazu.

Die anderen Baugruppen dienen dann der Ergänzung. So braucht man einen Bildschirm und eine Tastatur. Und natürlich Programme, die wir in einem getrennten Kapitel ausführlich besprechen.

4.1 Aufbau der SBC2-Computers

Ein Computer besteht ganz allgemein aus einer sogenannten Zentraleinheit, einem Speicher, einer Eingabeeinheit und einer Ausgabeeinheit. Dies gilt für jeden Computer.

Die Zentraleinheit führt die Arbeitsschritte aus. In dem Speicher können Befehle stehen, die an die Zentraleinheit gegeben werden oder Daten, die er z. B. für Berechnungen braucht. Über die Eingabeeinheit werden dem Computer Informationen aus der Umwelt gegeben. Die Eingabeeinheit kann eine Tastatur sein, aber auch aus Schalter, Tasten oder Fühler bestehen.

Bildschirm, Lampen, Motoren oder Drucker und Plotter dienen als Ausgabeeinheit.

4.1.1 Die erste Aufbaustufe: Startlogik und Taktgenerator

Abb. 4.1.1 zeigt schematisch die Elemente, aus welchen unsere Verarbeitungseinheit besteht. Da gibt es:

1. Eine Startlogik. Sie hat die Aufgabe, den Rechner zum Beispiel nach dem Einschalten zu starten. Und zwar muß der Rechner da mit einer ganz bestimmten Einstellung zu arbeiten anfangen.
2. Einen Taktgeber. Er liefert den Arbeitstakt für den Rechner. Ein Rechner muß nämlich seine Befehle in einem genau festgesetzten Rhythmus abarbeiten. Dazu benötigt er einen stabilen Takt. Je höher dieser Takt ist, desto schneller ist der Rechner bei seiner Arbeit. Ist der Takt zu hoch, so werden allerdings seine Schaltkreise überfordert. Jeder Rechner besitzt daher eine genau festgesetzte Takt-Frequenz, die für ihn optimal ist.
3. Die Zentraleinheit. Das ist der eigentliche Rechnerbaustein. Er wird oft CPU genannt, das ist die Abkürzung für Central Processing Unit, also auf deutsch: Zentraleinheit. In diesem Baustein laufen alle Operationen ab, die das Wesen eines Rechners ausmachen.

Zentraleinheit

Startlogik

Taktgeber

Abb. 4.1.1 Im Herzen des Computers arbeitet die Zentraleinheit (CPU). Sie wird von einem Taktgeber angetrieben und von einer Startlogik richtig gestartet

Als Zentraleinheit wird hier der Baustein mit der technischen Bezeichnung Z80-A verwendet. Der Buchstabe A gibt an, daß die CPU mit 4 MHz arbeiten kann. Neben den A-Bausteinen gibt es auch B-Bausteine (6 MHz), L-Bausteine (7,5 MHz) und H-Bausteine (8 MHz). Die Arbeitsfrequenz bestimmt, wie schnell Befehle ausgeführt werden können und damit, wie schnell Berechnungen durchgeführt werden können. Im folgenden wird der Buchstabe A stets weggelassen, da die weiteren Angaben unabhängig vom jeweiligen Typ sind.

Der Z80-Baustein, das ist also ein Vertreter der berühmten Mikroprozessoren. Es ist ein sehr bewährter Typ, der in der Industrie sehr häufig eingesetzt wird. Eine Platine mit dieser CPU sei jetzt Schritt für Schritt aufgebaut.

Erster Schritt: Wie funktioniert die Startlogik?

Zum einen soll der Rechner richtig starten, sobald die Spannung eingeschaltet worden ist. Zum anderen muß man ihn per Taste neu starten können, wenn man den Rechner dazu bringen will, daß er eine einmal begonnene Rechnung abbrechen soll und von vorn neu starten soll. Dazu wird ein mechanischer Taster verwendet, dessen Schaltsignal ausgewertet wird. Der Z80, unsere Zentraleinheit, besitzt einen separaten Eingang für solch ein „Rücksetz-Signal", der allerdings nicht direkt mit einem mechanischen Taster verbunden werden kann. Das hat folgenden Grund: Wenn ein mechanischer Taster betätigt wird, so berühren sich dessen Kontaktflächen mehrere Male kurz hintereinander. Man sagt, der Taster prellt.

Dieses Prellen entsteht, weil die Kontaktzungen wie Federn wirken und beim Aufeinanderprallen zu schwingen anfangen. Die Zeitdauer eines solchen Prellvorgangs liegt in der Größenordnung von Millisekunden (1 ms = $\frac{1}{1000}$ S), wobei noch die Bauart des Tasters eine Rolle spielt. Der Z80 würde bei einem solchen Signal an seinem Rücksetzeingang sehr durcheinander geraten. Daher muß eine Zusatzschaltung, die Startlogik, den Taster ergänzen. Diese Startlogik soll, wie schon gesagt, dafür sorgen, daß auch nach dem Spannungseinschalten erst einmal ein Startsignal an die Zentraleinheit gegeben wird.

Als erstes wollen wir also den Taster entprellen, also dafür sorgen, daß beim Betätigen nur ein Signalwechsel erfolgt. *Abb. 4.1.2* zeigt eine Schaltung, bei der ein Anschluß des Tasters mit dem 0-V-Anschluß (Masse) verbunden ist. Der andere Anschluß ist an einen Widerstand (R1) und an einen Kondensator (C1) geführt. Der Widerstand ist mit seinem anderen Ende an + 5 V angeschlossen und der Kondensator an 0 V (Masse).

Im Ruhezustand, also wenn die Taste nicht betätigt ist, kann sich der Kondensator C1 über den Widerstand R1 auf + 5 V aufladen. Wird der Taster nun betätigt, so wird der Kondensator sehr schnell entladen, und die Ausgangsspannung an Punkt A geht auf 0 V zurück. Wenn nun die Kontakt-Zungen des Tasters hin- und herfedern, so bedeutet das: die Taste wird praktisch ein paarmal kurz losgelassen. Der Kondensator verhindert, daß die Spannung am Kontakt in diesen

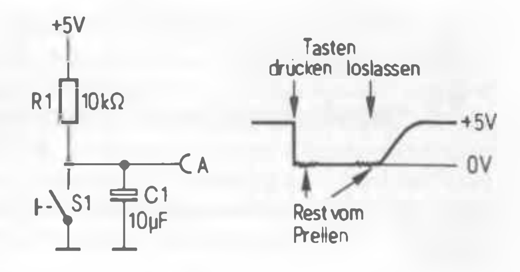


Abb. 4.1.2 Ein Taster, der mit einem RC-Glied entprellt wird. Ohne das RC-Glied würden im Moment der Betätigung des Tasters kurze Spannungssprünge zwischen 0V und 5V auftreten. Ebenfalls beim Loslassen des Tasters

kurzen Zeiten auf 5 V ansteigt; er wird über R1 so langsam aufgeladen, daß die Spannung praktisch ständig 0 V bleibt.

Auch beim Loslassen schließt der Taster noch ein paarmal nach dem ersten Öffnen. Wieder kann sich der Kondensator in dieser Zeit nicht auf + 5 V aufladen. Erst wenn das letzte Prellen aufgehört hat, wird er sich langsam auf + 5 V aufladen. Der Taster ist also mit einem R-C-Glied entprellt worden.

Nun hat aber diese einfache Schaltung in bezug auf den Z80 doch einen Haken. Das Signal steigt sehr langsam auf + 5 V an. Der Z80 zum Beispiel weiß mit solchen Signalen nicht viel anzufangen, seine digitalen Eingänge verlangen nach ihrer Konstruktion ein sehr schnell ansteigendes Signal, da sonst die interne Schaltung nicht richtig arbeitet. Wir müssen also unser Signal noch weiter aufbereiten.

Der nächste Schritt

Das langsam ansteigende Signal muß in ein schnell ansteigendes Signal umgewandelt werden. Dazu wird ein sogenannter Schmitt-Trigger verwendet. Dieser Baustein ist genau dafür konstruiert, sich langsam ändernde Signale so umzuformen, daß „steile“ Umschalt-Flanken entstehen.

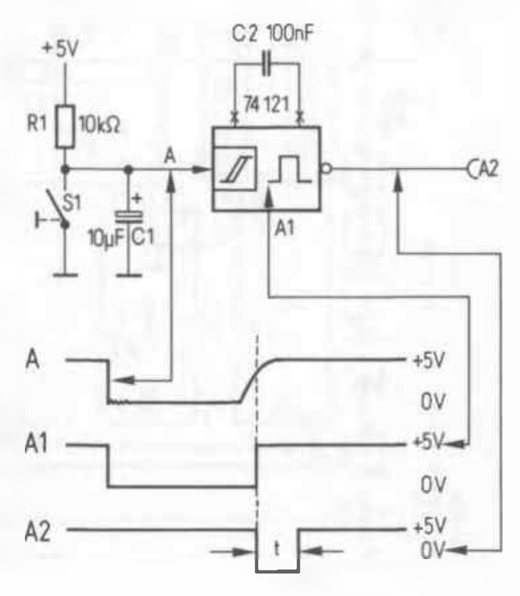


Abb. 4.1.3 Schaltung und Signal-Zeit-Plan der Startlogik. An Punkt A liegt das Signal aus Abb. 4.1.2 an. An Punkt A1 im Inneren des ICs könnte man das vom Schmitt-Trigger in ein sauberes "Rechteck" verwandelte Signal beobachten. An Punkt A2 liefert der im IC enthaltene "Monoflop" (deutsch etwa: Einmal-Impulsgeber) sein Ausgangssignal. Er wird dazu von der ansteigenden Flanke vom Signal A1 angestoßen (getriggert)

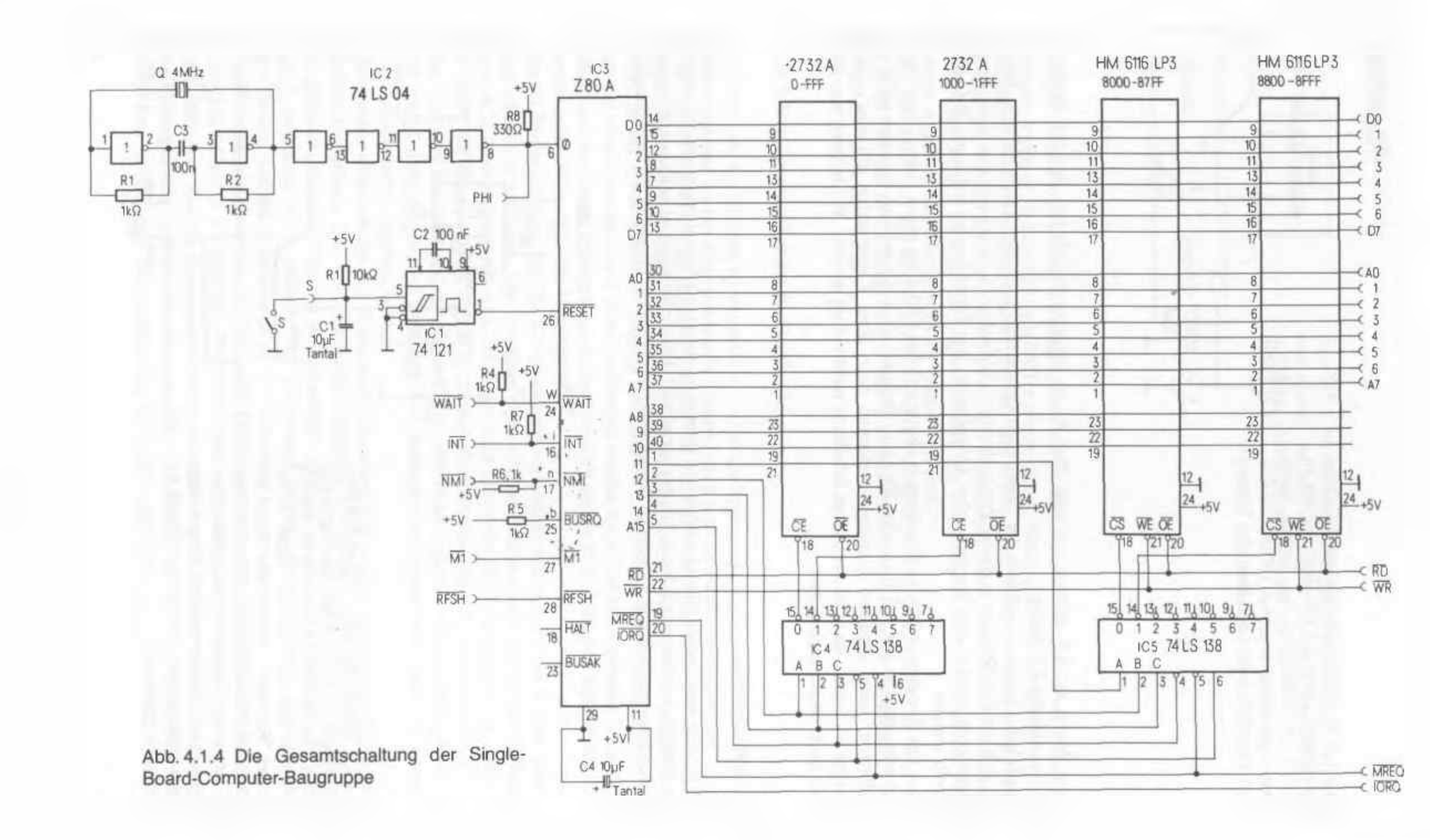


Abb. 4.1.4 Die Gesamtschaltung der Single-Board-Computer-Baugruppe

Im Bausatz gibt es dazu den integrierten Baustein mit der Typenbezeichnung 74121. Dieser Baustein enthält einen solchen Schmitt-Trigger. Er enthält aber auch noch ein anderes interessantes Element, nämlich ein sogenanntes Monoflop. Bitte lassen Sie sich jetzt von der Vielfalt an neuen Begriffen nicht verwirren: Die Ingenieure haben lange gebraucht, um sich das alles auszudenken. Wichtig ist, was das Monoflop tut. Es macht aus einem Eingangssignal einen immer gleich breiten Ausgangspuls. Genauer gesagt, wenn beim Monoflop im Inneren des 74121-Bausteins eine Signalflanke eintrifft, die von 0 auf 5 V ansteigt, dann antwortet es an seinem Ausgang mit einem Impuls, dessen Länge von außen genau einstellbar ist. *Abb. 4.1.3* zeigt die Schaltung und die Signalformen.

Das Signal A ist das Eingangssignal, das in den integrierten Schaltkreis 74121 eingegeben wird. In dem Schaltkreis sind ein Monoflop und ein Schmitt-Trigger in Reihe geschaltet. Am Punkt A1, der allerdings von außen nicht zugänglich ist, da er in der integrierten Schaltung verborgen liegt, ist das Signal nach der Bearbeitung durch den Schmitt-Trigger eingezeichnet. Am Punkt A2 tritt das Signal nach dem Monoflop auf. Es liegt normalerweise auf 5 V und geht, wenn der Taster losgelassen wurde, eine kurze, genau festgelegte Zeit auf 0 V. Die Zeitdauer, die im Bild mit t bezeichnet ist, wird durch einen Kondensator (C2) bestimmt. Dieses Signal kann nun direkt an die Zentraleinheit geführt werden. Von der Form her würde zum Betrieb der Zentraleinheit das Signal Al ausreichen. Jedoch würde der Z80 nicht arbeiten, solange der Taster gedrückt ist. Das würde später beim weiteren Ausbau stören. Daher wird das Monoflop verwendet, um den Z80 nur für kurze Zeit außer Gefecht zu setzen.

Abb. 4.1.4 zeigt den gesamten Schaltplan der SBC2-Baugruppe. *Abb. 4.1.5* zeigt den Bestückungsplan und *Abb. 4.1.6* die Stückliste. Erschrecken Sie nicht, es wird alles Schritt für Schritt aufgebaut und erklärt.

Abb. 4.1.7 zeigt die Bestückungsseite und *Abb. 4.1.8* die Lötseite des Layouts.

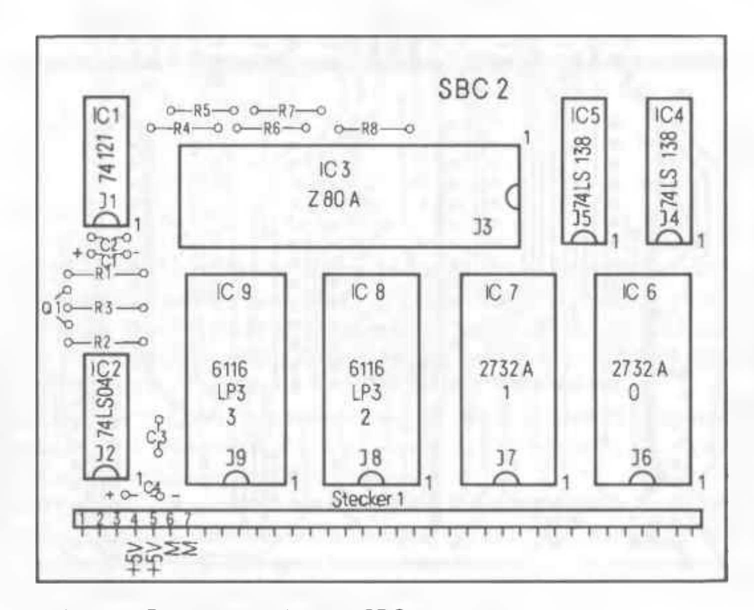


Abb. 4.1.5 Bestückungsplan der SBC 2

J1=IC1	74 121	
J2=IC2	74 LS 04	keinen 74 04 verwenden!
J3=IC3	Z 80 A	
J4=IC4	74 LS 138	
J5=IC5	74 LS 138	
J6=IC6	(2732 A, Grundprogramm, Basic oder Gosi)	
J7=IC7	(2732 A, Grundprogramm, Basic oder Gosi)	
J8=IC8	6116 LP3	
J9=IC9	6116 LP3	
R1	10 kΩ	
R2	1 kΩ	
R3	1 kΩ	
R4	1 kΩ	
R5	1 kΩ	
R6	1 kΩ	
R7	1 kΩ	
R8	300 kΩ	
C1	10 µF	
C2	100 nF	
C3	10 nF	
C4	10 µF	
Q1	4,000 MHz	
Stecker1	36polig	
S1	Taster	

Abb. 4.1.6 Die Stückliste zur SBC 2-Platine

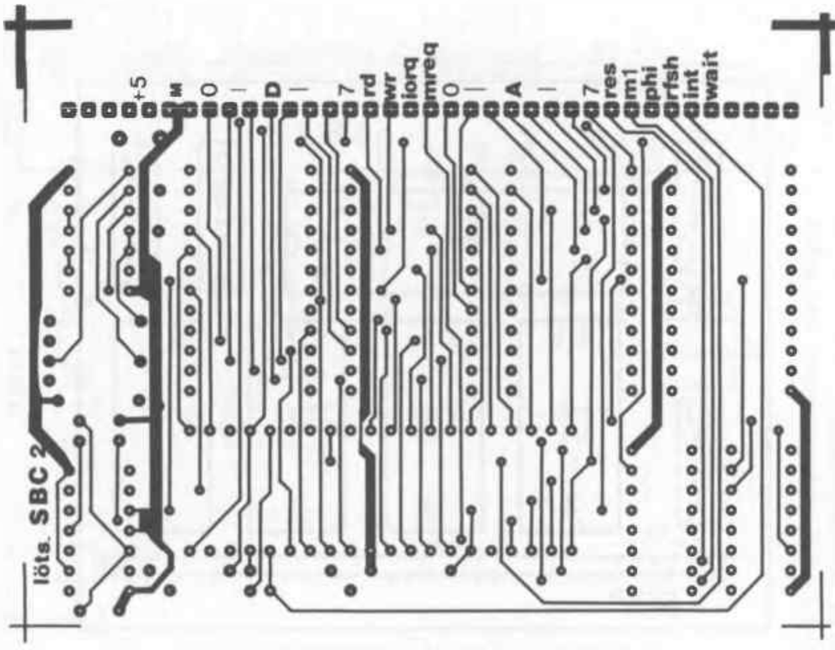


Abb. 4.1.7 Die Lötseite der Baugruppe SBC 2

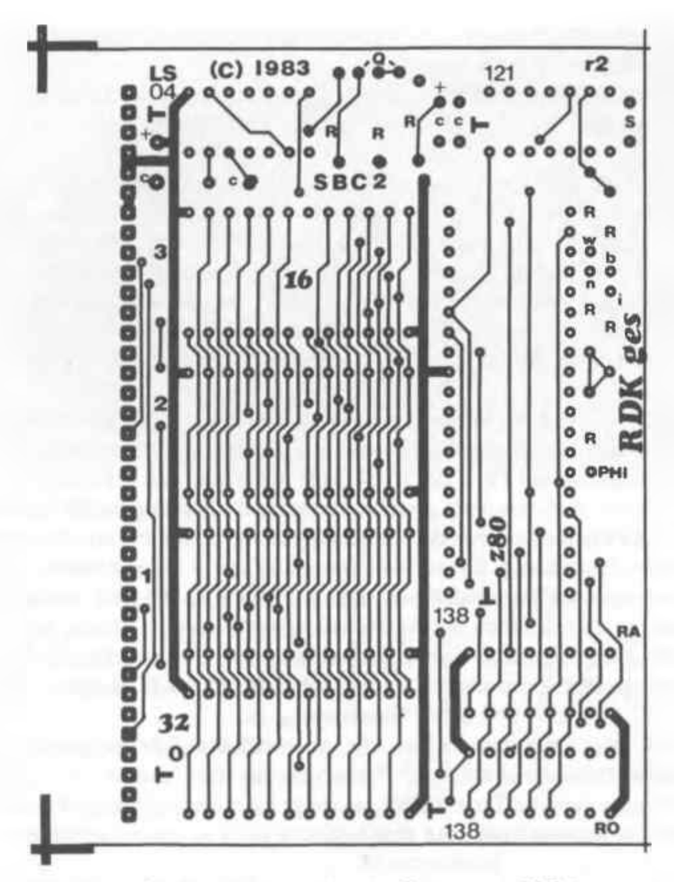


Abb. 4.1.8 Die Bestückungsseite der Baugruppe SBC 2

Jetzt wird gelötet

Erster Schritt: Wir löten den Sockel für IC1 ein. Achtung, so ein Sockel besitzt meist eine kleine Marke an einer der Schmalseiten. Diese Marke soll in Richtung auf die Steckerleiste zeigen. Den Sockel kann man nur sinnvoll einlöten, wenn man darauf achtet, daß er nicht auf der Lötseite, sondern auf der Bestückungsseite der Leiterplatte eingesteckt wird, so daß die Beine, die Pins, auf der Lötseite der Platine herausragen.

Diese Seite, auf der ausschließlich gelötet wird, ist mit „löts." am Platinenrand beschriftet.

Zweiter Schritt: Der Widerstand R1 wird eingelötet. Er besitzt einen Wert von 10 kΩ (Farbringe: braun-schwarz-orange-gold (oder silber).

Dritter Schritt: Der Kondensator C1 wird eingelötet. Es handelt sich um einen Tantalkondensator (Perlenform), manchmal auch um einen Elektrolytkondensator, mit einem Wert von 10 µF. Dabei ist entweder der Pluspol oder der Minuspol des Kondensators gekennzeichnet, und man muß beim Einbau unbedingt darauf achten, daß die Polung stimmt: Plus zur Plusmarkierung oder Minus zur Minusmarkierung!

Vierter Schritt: Der Kondensator C2 wird eingelötet. Dabei handelt es sich um einen 100-nF-Kondensator, bei dem die Polung keine Rolle spielt.

Fünfter Schritt: Ein Taster wird über zwei isolierte Litzen mit der Leiterplatte verbunden.

Sechster Schritt: Die 36polige Stiftleiste wird auf der Bestückungsseite eingesteckt und eingelötet.

Siebter Schritt: Nun wird die Spannungsversorgung POW5V mit der SBC2 gekoppelt. Man kann dazu die noch nicht geschilderte Bus-Baugruppe verwenden oder in diesem Fall auch einfach zwei Leitungen anlöten. Dabei auf keinen Fall vorn auf die Stifte der Stiftleiste löten, da man diese sonst später nicht mehr stecken kann. Man sollte die Stromversorgungszuleitungen auf der Lötseite der Leiterplatte anlöten. Dazu wird der + 5-V-Ausgang von POW5V mit dem + 5-V-Eingang der SBC2-Leiterplatte verbunden. Ebenso der 0-V-Ausgang mit dem 0-V-Eingang (Masse) der SBC2-Leiterplatte. (Das sind jeweils zwei Lötaugen mit der Bezeichnung + V und M.)

Achter Schritt: Nun wird die Versorgungsspannung eingeschaltet. Mit einem Vielfachinstrument (oder wahlweise Oszilloskop) mißt man die ankommende Spannung. Dabei wird der Minuspol des Instruments mit Pin 7 des IC-Sockels verbunden und der Pluspol mit Pin 14. Die Zählweise ist dabei so, daß, von oben gesehen, wenn die IC-Fassung mit der Markierung auf den Betrachter zeigt, der Pin rechts neben der Markierung Pin 1 ist. Der Pin mit der höchsten Nummer liegt dann links der Markierung. Es muß eine Spannung von + 5 V ankommen. Ist das nicht der Fall, so liegt entweder ein Kurzschluß vor, dann leuchtet auch die LED auf der POW5V nicht mehr; oder man hat versehentlich die Anschlüsse verpolt (dann wird eine negative Spannung angezeigt); oder es liegt irgendwo eine Unterbrechung vor, dann leuchtet die LED, aber keine Spannung wird angezeigt; oder man mißt am falschen Punkt des IC-Sockels, dann schaue man sich einmal die IC-Belegung und deren Numerierung an.

Neunter Schritt: Nun kann man noch an Pin (Anschlußbein) 5 des Sockels IC1 messen. Dort muß das entprellte Tastersignal anliegen. Wenn man die Taste drückt, so sinkt die Spannung schnell auf 0 V, wenn man die Taste losläßt, so steigt sie etwas langsamer wieder auf + 5 V an. Mit dem Vielfachinstrument kann man dies natürlich nicht so genau verfolgen wie mit einem Oszilloskop.

Zehnter Schritt: Spannung wieder ausschalten. Nun wird der integrierte Schaltkreis eingesetzt. Dabei muß man wieder auf die Orientierung achten. Auf dem integrierten Schaltkreis ist eine Marke auf der Schmalseite angebracht. Diese Marke muß bei IC1 zur Steckerleiste hinzeigen, also mit der des Sockels übereinstimmen *(Abb. 4.1.5)*. Wenn man sich nicht sicher ist, sollte man mal in Kapitel 3 nachsehen. Die Beschriftung des ICs gibt keine Anhaltspunkte auf die Orientierung. Sind zwei Marken auf dem IC vorhanden, so ist die *größere* Marke entscheidend.

Elfter Schritt: Die Spannung wird wieder angelegt. An Pin 1 des IC1 muß der entprellte kurze Puls ankommen, wenn man die Starttaste drückt. Diesen Puls kann man mit einem Prüfstift messen. Dazu wird dieser ebenfalls mit der Spannungsversorgung verbunden (zuvor abschalten) und dann der Prüfeingang an Pin 1 des ICs 74121 angelegt. Wenn man die Taste drückt, so muß jeweils genau ein Wechsel der LEDs von W1 auf W2 oder umgekehrt erfolgen. Dann arbeitet die Schaltung. Der Puls ist so schmal, daß man ihn mit dem Skop nur sehr schwer erkennen kann. Daher ist hier der Prüfstift das beste Testmittel.

Jetzt kommt der Taktgeber dran

Ein Taktgeber ist nötig, weil ein Computer in Schritten arbeitet, beinahe wie ein Uhrwerk. Der Taktgeber teilt der Zentraleinheit mit, wann und in welcher Geschwindigkeit die einzelnen Verarbeitungsschritte auszuführen sind.

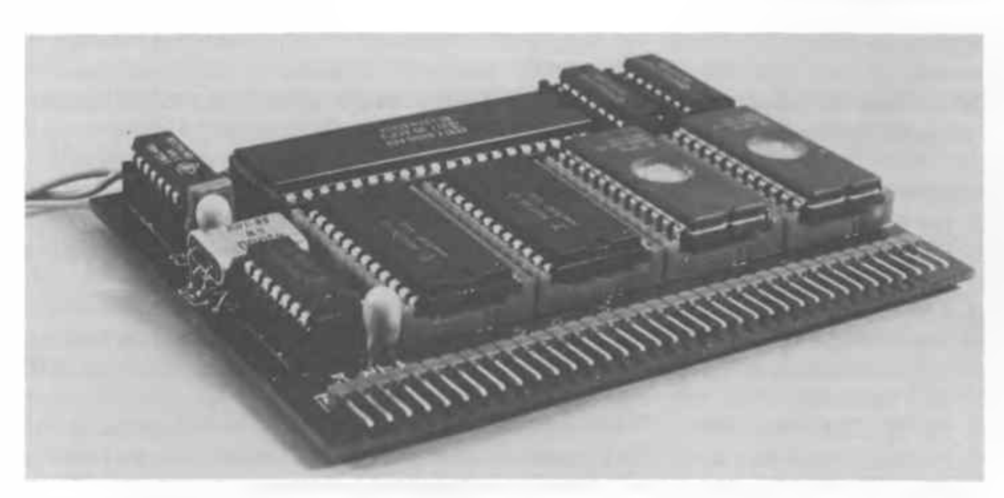


Abb. 4.1.9 So sieht die fertig aufgebaute Platine aus

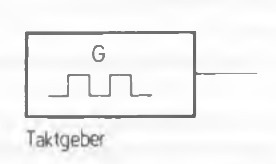


Abb. 4.1.10 Schaltsymbol für einen Taktgeber

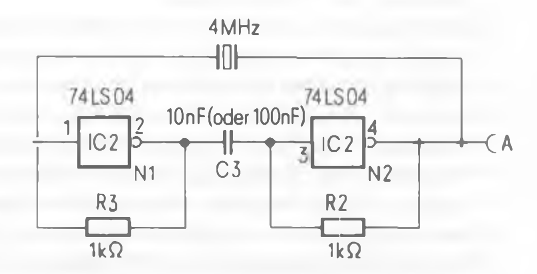


Abb. 4.1.11 Die hier verwendete Schaltung zur Takterzeugung

Kernstück des Taktgebers ist ein Schwingquarz. Dieser Schwingquarz liefert, betrieben mit der richtigen Beschaltung, eine sehr konstante Schwingung. Die Schaltung rund um den Quarz besteht ebenfalls aus einem integrierten Bauteil.

Abb. 4.1.11 zeigt den inneren Aufbau der Schaltung. Es zeigt zwei Nicht-Glieder (N1 und N2), einen Kondensator (C3), zwei Widerstände (R2 und R3) und einen Quarz Q.

Nehmen wir einmal an, am Eingang von N1 (Pin 1) liegt ein 0-Signal vor, also 0 V. Dann invertiert das Nicht-Glied diesen Wert und liefert am Ausgang eine 1, also + 5 V (bei ICs kann dieser Wert auch kleiner sein, minimal 2,5 V). Diese Spannung gelangt nun über den Widerstand R3 wieder zurück an den Eingang des Nicht-Gliedes. Diesmal aber als 1-Signal. Das wird nun wieder invertiert, und am Ausgang liegt ein 0-Signal an, das wieder an den Eingang gelangt und so weiter und so fort. Durch die Schwingungen des Nicht-Glieds wird nun auch der Quarz angeregt, seinerseits zu schwingen. Der Quarz kann aber (genau das ist sein Zweck) nur auf einer bevorzugten Frequenz schwingen. Daher kontrolliert der Quarz die Schwingungen des Nicht-Gliedes. Das Ergebnis ist ein stabiler Takt.

Das Nicht-Glied N2 arbeitet synchron mit N1 und gibt das Taktsignal weiter. Quarzfrequenzen gibt man meist in MHz an. Unser Quarz soll eine Frequenz von 4 MHz (vier Millionen Schwingungen pro Sekunde) liefern. Man könnte auch einen anderen Quarz verwenden, zum Beispiel 2 MHz. Dann arbeitet die CPU (Zentraleinheit) aber langsamer.

Und wieder Löten

Wir wollen die Schaltung nun aufbauen. Im Gesamt-Schaltbild *Abb. 4.1.4* findet man die Taktgeber-Schaltung wieder. Zum Aufbau:

1. Sockel des IC2 74LS04 einlöten und dabei auf die Marken achten.
2. Die beiden 1-kΩ-Widerstände R2 und R3 einlöten (Farbringe: braun-schwarz-rot).
3. Den Kondensator C3 einlöten. Als Werte sind 10 nF oder 100 nF verwendbar. Der Kondensator besitzt keine Polung. Daher spielt es keine Rolle, wie man ihn einlötet.
4. 4-MHz-Quarz einlöten. Der Quarz wird liegend eingebaut *(Abb. 4.1.9)*.
5. Das IC 74LS04 wird in die Fassung IC2 eingesteckt. Hier unbedingt wieder auf die Orientierung (Markierung IC und Markierung des Sockels) achten, denn ein falsch eingestecktes IC wird mit Sicherheit zerstört.
6. Die Spannung einschalten.
7. Mit dem Prüfstift wird an IC2, Pin 8, gemessen. Es müssen alle vier Leuchtdioden des Prüfstifts (H, L, W1 und W2) aufleuchten. Dann ist die Schaltung in Ordnung.
8. Man kann die Messung mit einem Oszilloskop auch genauer durchführen. An Pin 8 muß eine Frequenz von 4 MHz anliegen.
   Die Signalform ist dabei nicht so sehr entscheidend. Bei 4 MHz wird man kein exaktes Rechtecksignal mehr erhalten, da die Frequenz sehr hoch ist.
9. Fehlersuche: Wenn sich nichts tut, kann man, wenn man in einer Arbeitsgruppe arbeitet, das IC probeweise einmal mit dem des Nachbarn tauschen.
   Wird das IC sehr heiß, so wurde es wahrscheinlich falsch herum eingesteckt. Man kann dann versuchen, das IC nochmals richtig einzusetzen, meist ist es jedoch zerstört. Daher gut aufpassen beim Einsetzen von integrierten Schaltungen.
   Wenn die Schaltung bei korrekt eingesetztem IC nicht arbeitet, so kontrolliere man einmal die Lötseite der Leiterplatte. Meist liegt dann ein Lötfehler vor. Man kann zum Beispiel zwei Anschlußbeinchen versehentlich miteinander verlöten.

4.1.2 Die Zentraleinheit wird eingesetzt

Die SBC2-Baugruppe ist im vorherigen Abschnitt nicht fertiggestellt worden. Also weiter im Text:

Man löte alle restlichen Fassungen nach Bestückungsplan ein. Dabei auf die Markierungen achten!

Alle restlichen Widerstände (vier 1-kΩ-Widerstände mit dem Farbcode braun-schwarz-rot und 1 Widerstand 330 Ω mit dem Farbcode orange-orange-braun) jetzt einlöten.

Nun wird der Prozessor Z80-A feierlich in die 40polige Fassung eingesetzt. Bitte unbedingt auf die Orientierung achten. Und wenn es nicht gleich klappt, dann biegt man die Beine zurecht, indem man das IC in beide Hände nimmt, zwischen Daumen und Zeigefinger, und auf einer ebenen Unterlage mit sanftem Nachdruck alle 20 Beine einer Seite gleichzeitig biegt.

Spannung einschalten. Mit dem Prüfstift an Pin 6 der CPU messen. *Abb. 1* zeigt die Z80-CPU mit ihren Anschlüssen. Bei Pin 6 steht die Bezeichnung Takt. Dort muß also der 4-MHz-Takt des Taktgebers erscheinen. Beim Prüfstift leuchten alle vier LEDs (H, L. W1 und W2) auf, wenn alles korrekt funktioniert. Wenn man mit einem Oszilloskop arbeitet, so kann man auch nochmals die Frequenz überprüfen. Sie beträgt 4 MHz. Die Periodendauer beträgt also 250 ns (Nanosekunden).

Jetzt den Prüfstift an Pin 26 anschließen. Wenn man den RESET-Taster betätigt, so muß entweder die LED W1 ausgehen und W2 an oder W2 aus und W1 an. Da bis auf kurze Impulse ein 1-Signal an diesem Anschluß liegt, leuchtet die LED H. Der Eingang ist der RESET-Eingang der Zentraleinheit. Er ist direkt mit dem Ausgang der Start- und Rücksetzlogik verbunden.

Nun an Pin 11 messen. Dort muß die LED H leuchten, denn dort muß die + 5-V-Versorgung anliegen. An Pin 29 muß Masse anliegen und somit leuchtet die LED L, wenn man hier mit dem Prüfstift mißt.

Nun kann man den Prüfstift an die Anschlüsse 19, 20, 21 und 22 legen. Welcher Signal-Wert dort anliegt, hängt von den Umständen ab. Da auf der Platine noch wesentliche Bauelemente fehlen, sind die Reaktionen an den Ausgängen der CPU undefiniert.

Wenn man Bild 1 nochmals ansieht, so sind alle Anschlüsse der CPU mit Pfeilen versehen. Bei Ausgängen weisen sie von der CPU weg, bei Eingängen zur CPU hin. Dann gibt es aber auch noch acht Leitungen, die durch Doppelpfeile gekennzeichnet sind. Auf diesen Leitungen können Informationen sowohl in die CPU hinein als auch aus der CPU heraus transportiert werden.

Was ist ein Befehl?

Jetzt kommt etwas Wichtiges: Wenn der Taktgeber läuft und gerade am RESET-Eingang ein korrekter Impuls anlag, dann dauert es nur wenige Takte, bis die CPU an den Leitungen D0 bis D7 abtasten möchte, welche Pegel dort anliegen. Sie ist so gebaut, daß sie damit erfahren will, was sie als erstes tun soll. Sie will einen Befehl bekommen. Betrachten Sie nochmals *Abb. 4.1.12.* Die CPU erhält also alle Befehle über die sogenannten Datenleitungen. Diese Datenleitungen, die alle Doppelpfeile tragen, sind im Bild mit D0, D1, D2, D3, D4, D5, D6 und D7 bezeichnet. Wenn man alle diese Leitungen auf 0 V legt, so heißt das für die CPU: „tue nichts“. Für erste Experimente sei ein „Nichts-tu-Stecker“ aufgebaut. Man kann dazu einen Stecker in die 24polige Fassung stecken, in die später unser Speicher-IC kommt. Zuvor muß der Nichts-tu-Stecker aber noch geeignet verdrahtet werden. *Abb. 4.1.14* zeigt, wie. Man achte dabei auf das T-Symbol in der Zeichnung, das die Orientierung angibt. Links neben dem T-Symbol liegt Pin 1. Folglich werden, da man Pins immer gegen den Uhrzeigersinn zählt, die Pins 9, 10, 11, 12, 13, 14, 15, 16 und 17 miteinander verbunden. An Pin 12 liegt später Masse.

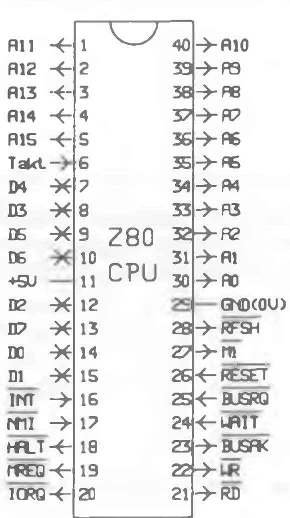


Abb. 4.1.12 Das sind die Anschlüsse der CPU, also der Zentraleinheit. Beachten Sie, daß alle Steuersignalbezeichnungen einen Querstrich tragen. Das bedeutet, daß sie bei Low-Pegel aktiv sind

Der Stecker wird dann (Achtung: Einbaulage beachten!) in die Fassung 0 der SBC2-Karte gesteckt. Man könnte für dieses Experiment auch jede andere Fassung verwenden, da die Datenleitungen an alle Fassungen geführt sind. Diese Datenleitungen sind über Leiterbahnen auf der Leiterplatte mit den entsprechenden Anschlüssen der CPU verbunden. D0 ist zum Beispiel an Pin 9 aller 24poligen Fassungen auf der SBC2-Baugruppe, D7 an Pin 17 zu finden. Man kann dies mit einem Durchgangsmesser prüfen (Vielfachinstrument auf Widerstandsmessung einstellen).

Immer wieder experimentieren

Wenn man die Spannung einschaltet und den RESET-Taster betätigt hat, sollte man folgende Messungen durchführen:

1. Mit dem Prüfstift an Pin 19 der CPU. Alle vier LEDs des Prüfstifts müssen leuchten.
2. Wir messen an Pin 21 der CPU. Auch hier müssen alle vier LEDs des Prüfstifts leuchten.
3. Wir messen an Pin 20 der CPU. Die LED H muß leuchten und dann noch W1 oder W2.
4. An Pin 22 darf nur die LED H leuchten und eine der LEDs W1 oder W2.

Mit dem Oszilloskop kann man die Signale genau ansehen. An Pin 19 und an Pin 21 erscheinen kurze Pulse, wie in *Abb. 4.1.13* sichtbar.

Was hat es mit diesen Leitungen auf sich? In *Abb. 4.1.12*, neben Pin 19, steht die Beschriftung $\overline{\text{MREQ}}$. Das bedeutet Memory-Request, zu deutsch Speicher-Anforderung.

Wenn die CPU auf dieser Leitung Pulse aussendet, so will sie etwas vom Speicher. Was sie genau will, sagt sie jedoch nicht allein mit diesem Signal.

Pin 21 ist mit $\overline{\text{RD}}$ bezeichnet. Das bedeutet Read, also Lesen. Wenn die CPU hier Pulse ausgibt, so will sie etwas lesen.

Jetzt wird auch die Bedeutung der Doppelpfeile an D0 bis D7 klarer. Wenn Pulse auf $\overline{\text{RD}}$ anliegen, so können in diesen Augenblicken Daten oder Befehle von außen über die Leitungen D0 bis D7 ins Innere der CPU gelangen.

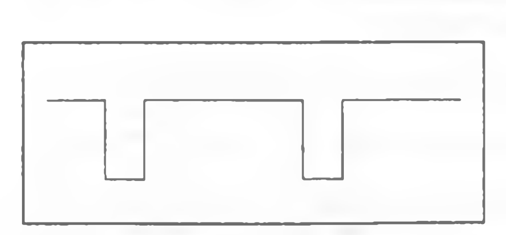


Abb. 4.1.13 In bestimmten Abständen tauchen auf der -RD-Leitung Impulse auf, die anzeigen, daß der Prozessor lesen möchte

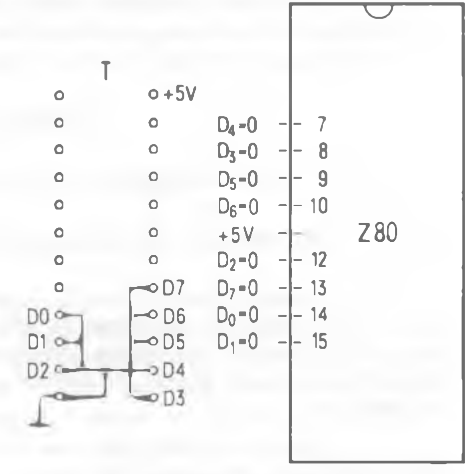


Abb. 4.1.14 Mit diesem Schaltungsvorschlag wird dem Z80 an seinen acht Datenleitungen konstant Low-Pegel, also 0, angeboten. Der Prozessor interpretiert das als Nichts-tu-Befehl, der in der Fachsprache NOP heißt

Bei Pin 22 steht die Beschriftung $\overline{WR}$. Das bedeutet Write, also Schreiben. Wenn hier Pulse anliegen, so will die CPU in diesen Augenblicken Daten über D0 bis D7 nach außen übertragen. Die Datenleitungen D0 bis D7 können also sowohl Daten von außen nach innen übertragen als auch umgekehrt. Fachleute sprechen in solch einem Fall von bidirektionalen Datenleitungen.

Nun noch zu Pin 20. Dort steht $\overline{IORQ}$. Das bedeutet Input/Output-Request. Gemeint ist, daß die CPU mit der Außenwelt in Verbindung treten will, diesmal aber nicht mit dem Speicher, sondern mit anderen Schaltungsteilen, die aus dem Computer heraus führen oder hinein. Man spricht von Peripherie.

Was die CPU tut, wenn sie nichts tut

Solange der Nichts-tu-Stecker im Sockel 0 steckt, wird man nur auf der $\overline{RD}$-Leitung und auf der $\overline{MREQ}$-Leitung Pulse feststellen. Klar, denn die CPU wollte weder etwas schreiben noch wollte sie etwas mit der Außenwelt zu tun haben, denn sie hatte ja aufgegeben bekommen, „nichts" zu tun.

Immer, wenn sie diesen Befehl bekommt, der aus lauter Lows (0 V) auf den Datenleitungen besteht, wartet sie eine genau definierte Anzahl von Takten und tut nichts weiter dabei. Dann fragt sie erneut nach einem Befehl und trifft in unserem Fall wieder auf den Nichts-tu-Befehl, der ja fest verdrahtet ist. Jedesmal wenn $\overline{RD}$ aktiv ist erfährt sie also, daß sie immer noch nichts tun soll.

Von Leitungsbezeichnungen und weiteren Befehlen

Mancher mag sich vielleicht über die Querstriche über den Bezeichnungen $\overline{MREQ}$ und anderen Leitungen wundern. Der Querstrich sagt, daß die Leitung im Ruhezustand ein 1-Signal führt. Bei $\overline{IORQ}$ und $\overline{WR}$ kann man auch ein 1-Signal mit dem Prüfstift (oder Oszilloskop) messen, solange der Prozessor nichts tut. Der Querstrich ist ein Hinweis des CPU-Herstellers, der die Orientierung erleichtern soll. Man sagt auch, daß solche Signalleitungen „low-aktiv" sind.

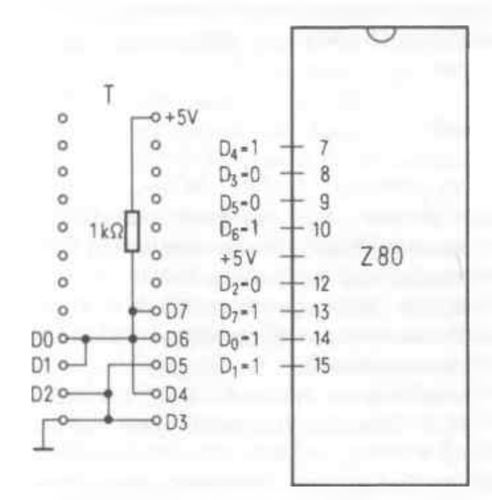


Abb. 4.1.15 Beim OUT-Befehl werden 11010011 auf den Datenleitungen erwartet

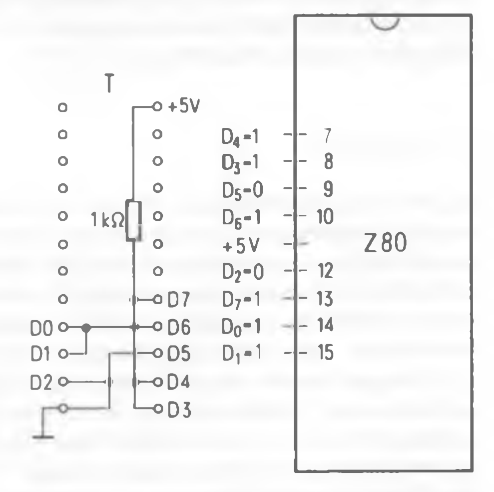


Abb. 4.1.16 Der IN-Befehl besteht aus dem Bitmuster 11011011

Wir können nun noch zwei weitere Versuche durchführen. *Abb. 4.1.15* und *Abb. 4.1.16* zeigen zwei weitere Verdrahtungen von Steckern.

Der Stecker nach *Abb. 4.1.15* ist so verdrahtet, daß er den OUT-(Output oder Ausgabe) Befehl an die CPU liefert. Mit diesem Befehl wird die CPU aufgefordert, Daten an die Außenwelt zu liefern. Es sollte jetzt schon immer klarer werden, daß die Zentraleinheit intern so aufgebaut ist, daß sie zu bestimmten Taktzeiten die Bitmuster, die ihr auf den acht Datenleitungen angeboten werden, übernimmt und intern als Befehl auswertet. Wenn man $\overline{IORQ}$, $\overline{WR}$, $\overline{MREQ}$ und $\overline{RD}$ mißt, so werden jetzt an allen vier Leitungen Pulse erkennbar sein (beim Prüfstift werden alle Leuchtdioden leuchten).

Der Stecker nach Abb. 5 ist ein IN-Stecker. Er veranlaßt die CPU Daten von der Außenwelt aufzunehmen. Bei einer Messung werden die Leitungen $\overline{IORQ}$, $\overline{RD}$ und $\overline{MREQ}$ Pulse zeigen und $\overline{WR}$ wird ein 1-Signal führen.

Hier ein Hinweis: Ein 1-Signal besteht beim Z80 meist nicht aus einem Signal von genau 5 V. Der genaue Wert ist sogar von IC zu IC verschieden. Er liegt zwischen + 2,5 V und 5 V. Ein 0-Signal liefert dagegen eine Spannung zwischen 0 V und 0,7 V.

Ein Prüfstift erkennt diese Pegel richtig. Mit dem Oszilloskop kann man die Sache genauer betrachten. Dabei zeigen sich oft noch merkwürdige Dinge auf dem Bildschirm: Den eigentlichen Pulsen sind oftmals kleinere Impulse überlagert. Für die Funktion sind sie ohne Bedeutung, solange die Gesamtspannung in den Spannungsbereichen 0 V bis 0,7 V und 2,5 V bis 5 V bleibt.

Wir fassen nochmals zusammen: Die CPU braucht Befehle zum Arbeiten. Diese Befehle gelangen über die Datenleitungen D0 bis D7 ins Innere der CPU. Die CPU besitzt spezielle Leitungen, über die sie den angeschlossenen Bausteinen mitteilt, ob sie Daten haben will oder welche ausgeben will, ob die CPU mit dem Speicher zu tun haben will oder mit der Außenwelt.

4.1.3 Dem Speicher auf der Spur

Die leeren Sockel auf der Platine SBC2 werden in Kürze mit höchstintegrierten ICs gefüllt werden. Es werden dort Speicherbausteine Platz finden, die dem Prozessor nicht nur NOP-Befehle mitteilen. Bevor das aber getan wird, muß einiges über Speicher gesagt werden, damit klar wird, was diese Bausteine tun.

Ordnung muß sein

Es gibt Leute, bei welchen es oben auf dem Speicher des Hauses wie „bei Hempels im Garten" aussieht. Dementsprechend werden diese Leute nichts wiederfinden. Andererseits gibt es Ordnungsfanatiker, die auch das geringste Stück etikettiert haben und darüber Buch führen, wo sie es aufheben. Mikrocomputer neigen ebenfalls zur Pedanterie. Allerdings etikettieren sie nicht das einzelne Stück, das sie aufheben wollen, sondern die Stellen, in welchen man etwas aufheben und wiederfinden kann. Jedem Speicherplatz ist eine Nummer zugeteilt.

Im vorigen Kapitel war als Übungsaufgabe die Überprüfung des Spieles auf den A-Leitungen gestellt worden. Sechzehn davon gibt es beim Z80. Dabei sollten Sie festgestellt haben, daß die Leitung A15 erwa im Sekundenrhythmus ihren Pegel wechselt, während A14 dies doppelt so geschwind tut und A13 dies viermal so schnell. Auch bei den Leitungen, bei welchen man das mit dem Prüfstift und dem bloßen Auge nicht mehr so einfach feststellen kann, ist es so, daß die mit der niedrigeren Nummer immer mit doppelter Frequenz „spielt", solange der NOP-Stecker

eingesteckt ist. Es passiert nämlich folgendes, nachdem der RESET-Taster betätigt wurde: Die CPU sendet auf allen 16-A-Leitungen O-Pegel aus, während sie gleichzeitig $\overline{\text{MREQ}}$ und $\overline{\text{RD}}$ betätigt. Mit den Nullen auf den A-Leitungen will sie einem angeschlossenen Speicher signalisieren, daß sie mit der Speicherstelle O etwas vorhat. Sie sehen schon, worauf das hinausläuft: Die A-Leitungen, das sind die Adreßleitungen, mit welchen die CPU durch Aussenden einer 16stelligen Binärzahl bestimmt, welche Speicherstelle angesprochen werden soll.

Aufgaben

1. Es soll ein Stecker gebaut werden, der das Datenmuster 01110111 an die CPU liefert. Dabei wird D7 und D3 mit 0 V belegt, der Rest mit +5 V. Achtung, man sollte die +5 V nie direkt anschließen, sondern immer über einen Widerstand, da sonst Kurzschlüsse entstehen, wenn die CPU Daten ausgibt. Ein Kurzschluß nach 0 V ist hingegen unschädlich. Der Stecker muß an $\overline{\text{RD}}$, $\overline{\text{MREQ}}$ und $\overline{\text{WR}}$ Pulse liefern, $\overline{\text{IORQ}}$ muß auf 1 bleiben.
2. Was tut der Stecker also?
3. Man sollte sich die anderen Leitungen der CPU ansehen. Wie ist die Reaktion bei verschiedenen Steckern?

Interessant ist beim NOP-Befehl das Spiel auf den Leitungen A0 bis A15. Während die A-Leitungen mit niedrigen Nummern ständig blitzschnell die Pegel ändern, wird dieses Spiel nach oben hin immer langsamer. Dort kann man den Wechsel mit bloßem Auge verfolgen.

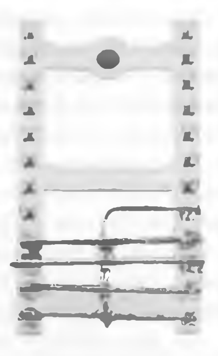


Abb. 4.1.17 So ist der Stecker für NOP aufgebaut. NOP ist die Abkürzung für No Operation. Solch ein Befehl ist sinnvoll, weil damit genau festgelegte Zeiten ohne Aktion überbrückt werden können

Betrug

Mit dem eingesteckten Nichts-tu-Stecker wurde die CPU deshalb betrogen, weil ihr nur Speicher vorgetäuscht wurde. Nach dem Aussenden der Adresse und dem Betätigen der beiden Steuersignale $\overline{MREQ}$ und $\overline{RD}$ erwartet die CPU mit blindem Vertrauen, daß jetzt der Speicher mit dem Inhalt der angesprochenen Speicherzelle antwortet. Sie kann sich nicht vorstellen, daß nur ein so merkwürdiger Stecker die Pegel auf den Datenleitungen verursacht hat. Sie interpretiert also das Angebotene als Befehl, wie schon geschildert, und führt ihn durch. Die CPU ist nun so aufgebaut, daß sie nach der Durchführung des NOP-Befehls erneut einen Befehl holen möchte. Dazu sendet sie jetzt auf den Adressen-Leitungen die nächste Speicherzellennummer aus, eine Eins in diesem Fall.

Befehlszyklen

Bevor wir genauer den Speicher betrachten, sei also betont, daß die CPU im Experiment aus dem vorigen Kapitel durch den Reset-Impuls zunächst intern in einen Zustand gebracht wird, von dem aus sie die Adresse 0 auf den Adreßleitungen ausgibt und die kommende Antwort als Befehl interpretiert. Nach der Absolvierung des ersten Befehles, der diesmal ein NOP ist, sendet sie die Adresse 1 aus und erwartet wieder einen Befehl, der wegen des Steckers wieder ein NOP ist. Nach diesem sendet die CPU, so ist sie nämlich konstruiert, die nächste Adresse aus, die 2., danach die 3. und so fort. Jedesmal täuscht der Nicht-tu-Stecker den Inhalt NOP vor und die CPU tut ein paar Takte nichts. Das Spiel läuft durch, bis alle 16 Adressen-Leitungen 1 sind und hält auch dann nicht inne, denn die CPU ist so aufgebaut, daß sie dann wieder bei 0 anfängt. Das Experiment aus dem vorigen Kapitel würde also ewig laufen, wenn man nicht den Strom irgendwann abschalten würde. Die Adressen würden dabei immer wieder von 0 bis $2^{16}-1$, das ist 65536–1, hochgezählt werden. Ebenso viele NOP-Befehle werden dabei absolviert. Vielleicht versuchen Sie einmal auszurechnen, wie lange die CPU für einen einzigen solcher Befehle benötigt.

Auswählen auf elektronisch

Die Z80-CPU ist also so konstruiert worden, daß sie einen Speicher von möglicherweise 65536 Speicherzellen erwartet, von welchen sie in bestimmten Situationen genau eine Zelle auswählen möchte, um zu erfahren, was darin steht.

Die an die CPU angeschlossenen Speicherbausteine müssen so konstruiert sein, daß sie die Signale der CPU verstehen und entsprechend reagieren.

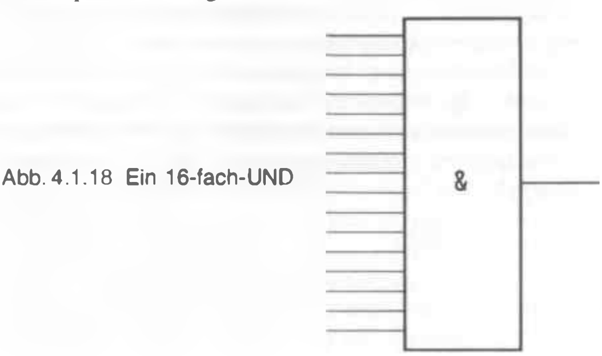


Abb. 4.1.18 Ein 16-fach-UND

Ein wichtiges Detail dabei ist die Umsetzung der 16 Signale auf den Adreß-Leitungen, damit genau die gewünschte Speicherzelle angesprochen werden kann. Das Interessante ist, daß das mit den elementaren Logik-Bausteinen aus dem ersten Kapitel schon gelingt. Die *Abb. 4.1.18* zeigt dazu zunächst einen UND-Baustein mit 16 Eingängen. Gewiß zunächst ein Unikum. Dieser Baustein antwortet an seinem Ausgang genau dann mit einer 1, wenn alle seine 16 Eingänge auf 1 liegen. Ihn könnte man also dazu benutzen, genau diejenige Speicherzelle zu aktivieren, die die Nummer 65 535 trägt, denn wenn eine oder mehrere Adreßleitungen 0 führen, ist auch das UND nicht aktiv. Es reagiert wirklich nur auf die Zahl 65 535 und auf keine andere.

Setzen Sie jetzt gedanklich vor alle UND-Eingänge einen Inverter. Diese neue Schaltung ist genau dann am Ausgang aktiv, wenn alle Eingänge 0-Signal führen. Mit solch einer Schaltung kann man also die nullte Zelle anwählen. Wenn beide Schaltungen gleichzeitig an denselben Adressenleitungen hängen würden, könnten also schon zwei Zellen exakt angesteuert werden.

Lassen Sie jetzt den Inverter, der an der Adreßleitung A0 hängt, aus der Schaltung weg, die die Null auswählen kann. Alle anderen behalten Sie bei. Diese Schaltung meldet sich genau dann, wenn die Binärzahl 1 auf den Adreßleitungen ausgesandt wird.

Wenn Sie den Inverter weglassen, der von Adreßleitung A1 bedient wird und den bei A0 ebenfalls, dann meldet sich diese Schaltung genau bei 3.

Ganz allgemein: Wenn Sie eine Schaltung bauen wollen, die sich genau bei einer bestimmten Binärzahl meldet, dann nehmen Sie ein UND mit so vielen Eingängen, wie die Binärzahl Stellen besitzt, und setzen genau dort Inverter vor den UND-Eingang, wo die Binärzahl an der entsprechenden Stelle 0 stehen hat. Bei den Stellen mit 1 kommt kein Inverter davor. Genau bei dem vorgegebenen Bitmuster meldet sich dann die Schaltung.

Die Decodierung

Stellen Sie sich jetzt eine Schaltung vor, in der 65 536mal 16-Fach-UNDs, nach dem vorhergegangenen Schema mit Invertern versehen, gleichzeitig an den 16 Adreßleitungen des Z80 hängen *Abb. 4.1.19*. Jedesmal, wenn der Z80 eine Binärzahl auf seine Adreßleitungen legt, meldet sich dann genau das eine UND an seinem Ausgang, das die Inverter an seinen Eingängen entsprechend verteilt hat. Man sagt, daß die eben vorgeschlagene Schaltung die Adressen des Z80 decodieren kann. Sie besitzt 65 536 Ausgänge, von welchen jeweils genau der aktiv ist, dessen Nummer als Binärzahl angeliefert wird. Mit dem Ausgangssignal könnte man also genau die gewünschte Speicherzelle freischalten. Diese Riesenschaltung ist allerdings so nirgendwo in einem Speicherbaustein realisiert, weil sie viel Aufwand bedeuten würde.

Im Bausatz für die SBC2-Baugruppe ist aber doch auch genau solch ein Decoderbaustein beigelegt, weil er an bestimmter Stelle noch benötigt werden wird. Genauso heißt hier, daß er nach dem geschilderten Prinzip intern aufgebaut ist, allerdings nur drei „Adressenleitungen" nach außen geführt hat und entsprechend nur 8 Ausgänge besitzt, die jeweils aktiv werden, wenn das entsprechende Bitmuster, also die entsprechende dreistellige Binärzahl an seinen dafür vorgesehenen Eingängen anliegt *(Abb. 4.1.20)*. Er besitzt noch eine weitere Besonderheit: Den acht dreistelligen UNDs in seinem Inneren ist jeweils ein Inverter nachgeschaltet, so daß ein Ausgang mit 1 inaktiv ist, während ein Ausgang mit 0 anzeigt, daß seine „Zahl" an den Eingängen anlag. Und diese UNDs besitzen noch einen vierten Eingang, über den sie gesperrt werden können. *Abb. 4.1.20* zeigt sein Schaltbild und seine Funktionstabelle. Daran kann man ablesen, daß neben den drei Eingängen A, B, C noch Steuereingänge G_1, G_{2A}, G_{2B} existieren, von welchen jeder die Ausgänge alle auf 1 bringen kann, wenn das entsprechende Potential an ihm liegt – und zwar unabhängig davon, was dann an den anderen Eingängen liegt.

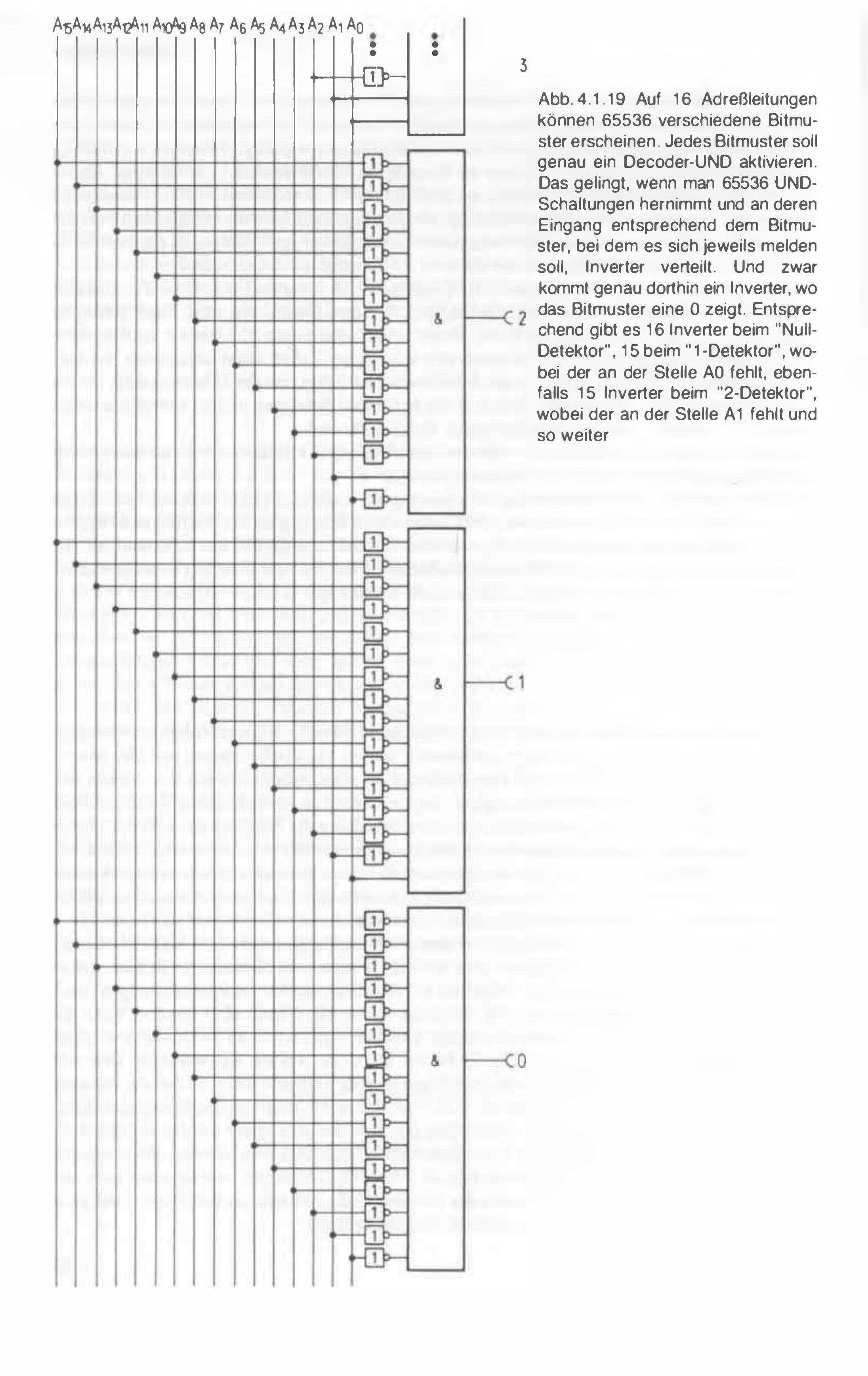
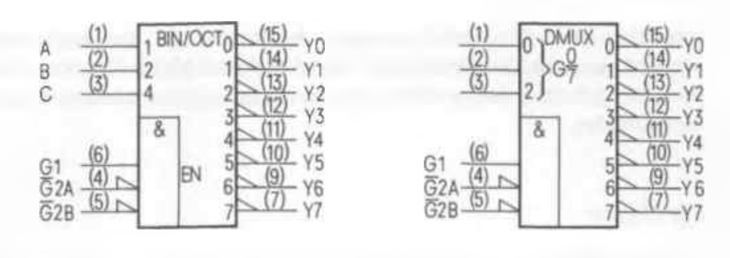


Abb. 4.1.19 Auf 16 Adreßleitungen können 65536 verschiedene Bitmuster erscheinen. Jedes Bitmuster soll genau ein Decoder-UND aktivieren. Das gelingt, wenn man 65536 UND-Schaltungen hernimmt und an deren Eingang entsprechend dem Bitmuster, bei dem es sich jeweils melden soll, Inverter verteilt. Und zwar kommt genau dorthin ein Inverter, wo das Bitmuster eine 0 zeigt. Entsprechend gibt es 16 Inverter beim "Null-Detektor", 15 beim "1-Detektor", wobei der an der Stelle A0 fehlt, ebenfalls 15 Inverter beim "2-Detektor", wobei der an der Stelle A1 fehlt und so weiter



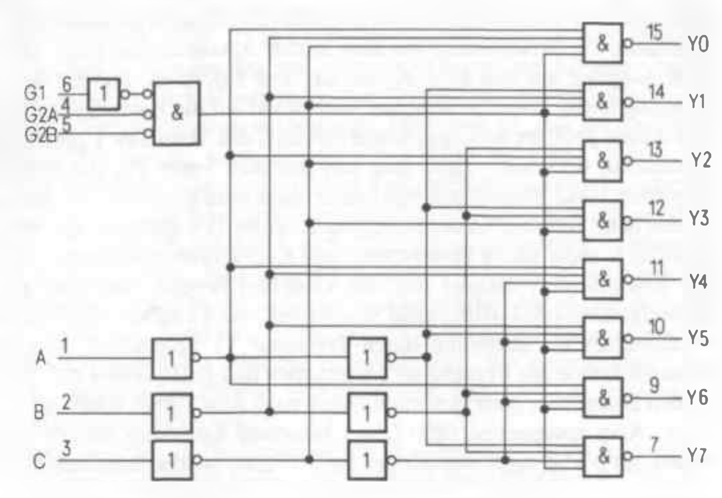


Abb. 4.1.20 Ein Dekoder, wie er dem Bausatz beiliegt. Oben die Schaltung symbolisch dargestellt, unten die Innenschaltung, wie sie im IC arbeitet

Es gibt noch eine Besonderheit: Nicht bei jedem UND sind hier die Inverter passend verteilt, sondern nach den Eingängen A, B, C werden das invertierte $\overline{A}$, $\overline{B}$, $\overline{C}$ und das nichtinvertierte Signal bereitgestellt und die Eingänge der UNDs werden passend an das invertierte oder nichtinvertierte Signal gelegt. Das spart sehr viele Inverter. Bitte betrachten Sie *Abb. 4.1.20* genau. Solche Schaltungen sind wichtig.

Ins Innere der Speicherbausteine

Es gibt verwirrend viele Speicherbausteine, sowohl, was deren interne Funktionsweise als auch deren technologischen Aufbau betrifft. Im Prinzip aber wird bei allen Bausteinen zunächst eine Adresse decodiert und dann je nach Steuersignal entweder ein Inhalt ausgegeben oder ein neuer Inhalt eingegeben. Stellen Sie sich vor, daß jeder Ausgang eines Decoder-UNDs ein Relais bedient, das acht Kontakte besitzt und damit acht Bit-Leitungen gleichzeitig nach außen durchschalten kann. Im Inneren des Speichers sitzen nun noch acht Vorrichtungen bei jedem der Relais, die jeweils 0- oder 1-Pegel auf die Bitleitungen legen können. Beispielsweise könnten das acht Schalter sein, die ein oder aus sind. Wenn also am Decodereingang eine Binärzahl anliegt, dann wird das entsprechende Decoder-UND aktiviert und so der Zustand der dahinterliegenden Speicherzelle, also das Muster der dort eingestellten 0- oder 1-Pegel, durch das Relais auf die Ausgangsleitungen durchgeschaltet. Natürlich ist das eben Gesagte heute nicht mehr modern,

zeigt aber exakt das Funktionsprinzip. Im Inneren der Speicherbausteine gibt es keine Relais, sondern nur noch Transistoren und manchmal auch kleine Kondensatoren. Sowohl die Relais als auch die Schalter, mit welchen die Bitmuster erzeugt wurden, sind aus solchen Transistoren nachgebildet.

Ein Flipflop

Damit Sie sich ein bißchen vorstellen können, wie in modernen Halbleiterspeichern einzelne Bits abgespeichert werden können, sei folgendes Experiment gemacht *(Abb. 4.1.21):* Sie nehmen die zweifache Treiberschaltung aus dem letzten Abschnitt und führen da einen 10-kΩ-Widerstand vom Ausgang auf den Eingang zurück. Der Effekt ist, daß bei nicht leuchtender Lampe am Ausgang hohes Potential über den Widerstand auf die Basis des Transistors am Eingang gelangt und diesen geöffnet hält, was weiter bewirkt, daß Transistor 1 geschlossen bleibt, weil er keinen Basisstrom bekommt. Tippt man nun mit dem freien Pin des ebenfalls am Eingang von T2 liegenden 1-kΩ-Widerstandes an Masse, dann wird wegen des Widerstandsverhältnisses von 1 zu 10 das hohe Potential an der Basis von T2 gegen 0 V gezogen, der Transistor bekommt an seiner Basis nicht mehr genug Steuerstrom, sein Kollektorstrom verringert sich ebenfalls, weshalb auch das Potential dort ansteigt und das wiederum bewirkt, daß über den dort zur Basis von T1 abzweigenden 1-kΩ-Widerstand Strom fließt, der T1 öffnet. Ergebnis: Das einmal Tippen an 0 V ist dauerhaft als Stromfluß durch Transistor T1 gespeichert. Umgekehrt kann man mit der Steuerelektrode am Eingang an 5 V antippen und schon öffnet T2, während T1 schließt, weil er keinen Basisstrom mehr bekommt. Auch nach dem Tippen bleibt die Schaltung in der jeweiligen Lage. Also zusammengefaßt: Diese Schaltung kann sich merken, ob mit dem Steuereingang zuletzt an 0 V getippt wurde oder an 5 V. Eine solche Schaltung wurde von den Ingenieuren

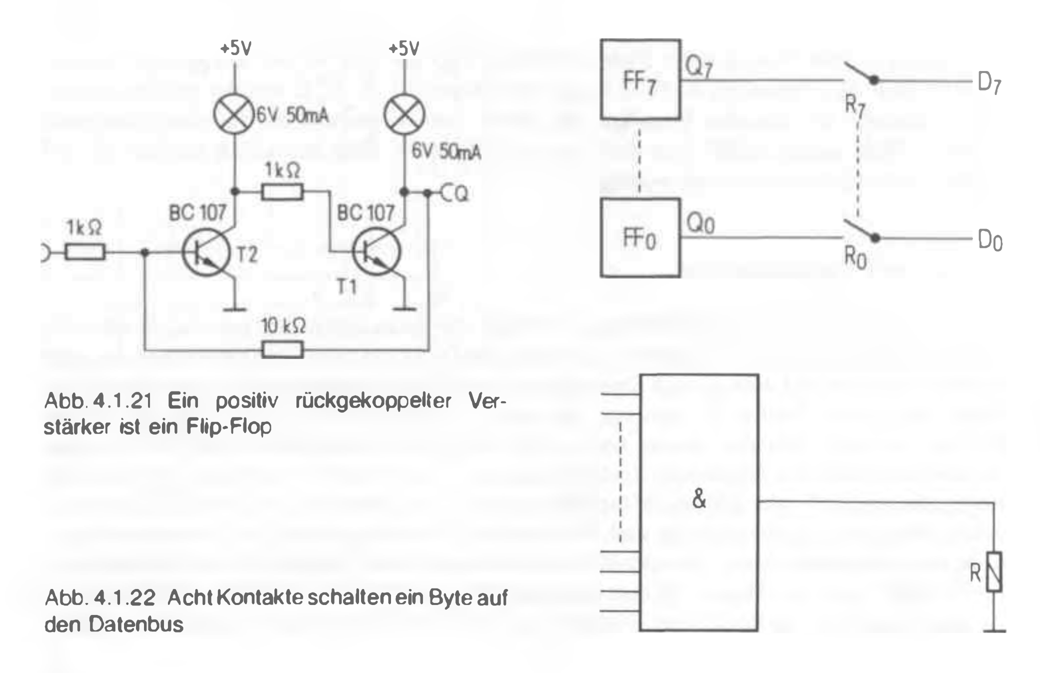


Abb. 4.1.21 Ein positiv rückgekoppelter Verstärker ist ein Flip-Flop

Abb. 4.1.22 Acht Kontakte schalten ein Byte auf den Datenbus

Flipflop getauft, nach dem Geräusch, das ein angeschlossener Lautsprecher macht, wenn umgeschaltet wird. Betrachten Sie diese Schaltung ein wenig mit Ehrfurcht. Neben dem Decoder ist sie (mit ihren modernen Varianten) wichtigstes Element in der Computertechnik.

Schreiben und Lesen

Das Flipflop selbst ist also eine der wichtigsten Schaltungstypen, denn es ist in vielen Varianten in CPUs und in Speicherbausteinen eingebaut. Denken Sie jetzt wieder an die Signale der Z80-CPU. Da gab es die Leitungen $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ und $\overline{\text{WR}}$. Diese Leitungen führen je nach Absicht der CPU Impulse. Wenn aus dem Speicher gelesen werden soll, dann wird neben der Adresse sowohl $\overline{\text{MREQ}}$ als auch $\overline{\text{RD}}$ aktiv. Diese beiden letztgenannten Signale werden nun, von oft schon in den Speicherbausteinen eingebauten Freigabelogiken ausgewertet. Und zwar so, daß der Speicherbaustein bei Vorliegen des $\overline{\text{RD}}$-Impulses, wenn gleichzeitig $\overline{\text{MREQ}}$ aktiv ist, einfach die momentane Stellung der acht Flip-Flops einer angewählten Speicherzelle nach außen „durchschaltet". Das heißt, der Pegel am Ausgang eines jeden der acht Flipflops des angewählten Bytes wird auf die Leitungen D0 bis D7 gebracht, um dort von der CPU registriert werden zu können.

Wenn die CPU etwas schreiben möchte, dann aktiviert sie den $\overline{\text{WR}}$-Ausgang, während $\overline{\text{RD}}$ inaktiv bleibt. In den Speicherbausteinen wird dieses Signal so ausgewertet, daß die auf den acht Datenleitungen von der CPU erzeugten Pegel jetzt genau den acht Eingangssteuerleitungen der acht von den Adressenleitungen angewählten Flipflops zugeführt wird, die genauso ihre Lage danach richten, wie das eine handgemachte Flipflop im Experiment, und diese Lage auch dann beibehalten, wenn alle Steuerimpulse vorbei sind.

Es sei nochmals betont, daß alle Dinge, die bisher geschildert wurden, eher die Funktionsprinzipien schildern, als den wirklichen Aufbau von Speicherzellen, denn dort haben sich im Zuge der Entwicklung der Technik noch höchst raffinierte Varianten herausgebildet, die nicht so leicht erkennen lassen, auf welch einfachen Tatsachen alles beruht.

Die verschiedenen Speichertypen

Abb. 4.1.23 zeigt eine Reihe von Speichertypen. Wir betrachten zunächst den Speicher mit der Beschriftung RAM. RAM bedeutet Random Access Memory oder auf deutsch: Speicher mit wahlfreiem Zugriff. Das soll anzeigen, daß man Informationen in diesem Speicher einschreiben und auch wieder auslesen kann, genauso, wie vorhin geschildert. Solche ICs besitzen einen Eingang, der mit „Aktivieren" beschriftet ist. Liegt dort ein Signal an, so wird der Speicher in Arbeitsbereitschaft versetzt. Erst jetzt achtet er auf Signale an seinen anderen Anschlüssen.

Dann gibt es einen Dateneingang. Der muß nicht immer 8 Bit breit sein, meist ist er sogar nur 1 Bit breit. Dort wird die Information angelegt, also zum Beispiel, ob ein 1-Signal oder ein 0-Signal gespeichert werden soll. Ferner findet man einen Eingang „Schreiben". Erscheint dort ein Puls, so wird die Information am Dateneingang in den Speicher übernommen. Wenn man Informationen auslesen will, so legt man einen Puls an den Eingang „Lesen". Am Datenausgang erscheint dann die Information.

Dann gibt es noch einen Eingang mit der Beschriftung „Adresse". Dahinter verbergen sich die Decoderelemente. Bei den Speicherbausteinen in der Wirklichkeit werden die Speicher-Flipflops in einer Anzahl von Spalten angeordnet.

Um eine Speicherzelle zu „adressieren", werden intern eine Spalte und darin eine Reihe angesprochen (engl. columns and rows). Eine einzelne Speicherzelle wird nur aktiviert, wenn die

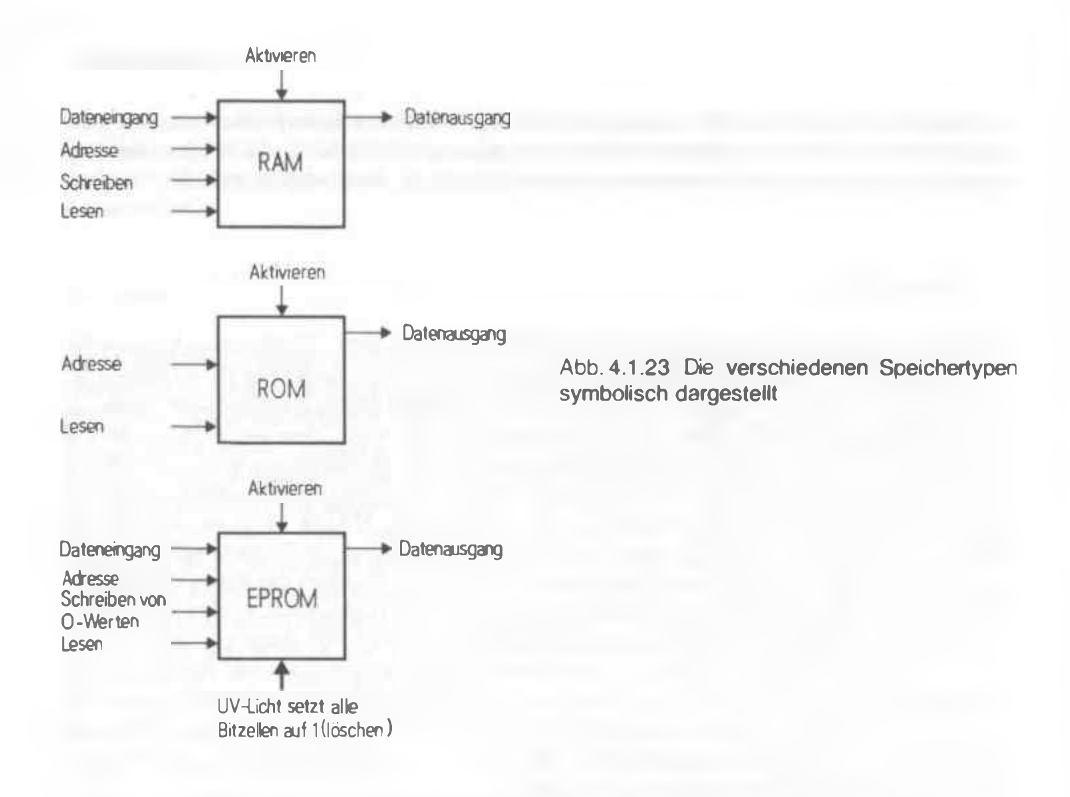


Abb. 4.1.23 Die verschiedenen Speichertypen symbolisch dargestellt

Spaltenleitung und die Zeilenleitung aktiviert sind. Durch die Matrixanordnung wird die Zahl der benötigten UNDs im Decoder von n auf $2 \times \sqrt{n}$ reduziert. Beispiel: Gegeben sind 1000 Speicherzellen, dann sind also 20 UNDs nötig, somit je 10 Reihen und 10 Spalten. Denn $10 \times 10 = 100$.

Ein anderes Beispiel: Ein Speicher faßt 65 536 Bit. Wie viele Adreßleitungen braucht man? Den Speicher kann man in 256×256 Speicherzellen aufteilen, also hat man je 256 UNDs bei den Spalten und bei den Reihen zu codieren. Dazu benötigt man je 8 Dualstellen, gesamt also 16 Dualstellen. Mit 16 Adreßleitungen kann man also eine Speicherzelle aus diesem Speicher auswählen.

Vom EPROMs und ROMs

In *Abb. 4.1.23* sind noch andere Speicherbausteine gezeigt. Da gibt es zum Beispiel sogenannte ROMs. Ihnen fehlt der Schreibeingang und der Dateneingang. Man kann aus diesen Bausteinen nur Daten lesen. ROM ist die Abkürzung für Read Only Memory, zu deutsch „Nur-Lese-Speicher".

Bei ROMs geschieht der Einspeichervorgang schon bei Herstellung der integrierten Bausteine. Durch einen metallischen Aufdampfungsprozeß werden die Informationen (oft durch gezieltes Schließen oder Offenlassen einer Bitleitung wie bei einem Schalter) fest vorgegeben. ROMs werden immer dann verwendet, wenn zum einen die Informationen dauerhaft sein sollen, und zum anderen sehr große Stückzahlen verwendet werden sollen, denn der metallische Aufdampfungsprozeß ist teuer, wollte man nur einzelne ICs herstellen. Wenn man Einzelstücke mit festem Inhalt benötigt, so verwendet man die sogenannten EPROMs, Eraseable Programmable Read Only Memories. Auf deutsch: Löschbare Programmierbare Nur-Lese-Speicher.

Diese EPROMs haben wieder einen Schreibeingang, jedoch kann man damit nur 0-Signale einschreiben, wenn also im Speicher schon ein 0-Signal gespeichert war, so kann man daraus kein 1-Signal mehr machen. Dazu gibt es aber im Deckel des EPROMs ein Quarzfenster. Wenn man das EPROM durch dieses Quarzfenster mit UV-Strahlen belichtet, so werden alle Speicher-Zellen in den 1-Zustand überführt. Dieser Prozeß dauert etwa eine viertel Stunde, EPROMs sind also empfindlich gegenüber Licht. Tageslicht ist in der Lage, EPROMs innerhalb weniger Stunden zu löschen, also 1-Signale einzuspeichern. Wenn man EPROMs verwendet, sollte man also besser den Quarzdeckel mit einem Aufkleber versehen und so das IC vor Licht schützen.

Das Einschreiben von Informationen in EPROMs hat noch eine Besonderheit. Es dauert relativ lange. Während man Daten im Bereich 200 ns (Nanosekunden, 1 ns = $\frac{1}{1000000000}$ s) auslesen kann, braucht man 50 ms (Millisekunden 1 ms = $\frac{1}{1000}$ s) zum Einschreiben der Daten in eine einzelne Speicherzelle. Daher verwendet man EPROMs als ROM-Ersatz, wenn es darum geht, Daten oder Programme dauerhaft zu speichern. Leider kann man EPROMs nicht beliebig oft löschen, sie lassen sich irgendwann einmal nicht mehr programmieren. Wie wird die Information in einem EPROM gespeichert?

Das ist ganz trickreich. Im EPROM befinden sich speziell gebaute Kondensatoren. Auf diese Kondensatoren wird beim Einschreiben eine Ladung gebracht. Diese Ladung bestimmt, ob ein 1-Signal gespeichert ist (keine Ladung da) oder ein 0-Signal (Ladung da). Durch die UV-Strahlung wird die Ladung wieder entfernt und damit der Speicher mit 1-Werten belegt.

Dynamische RAMs

Bei RAM-Bausteinen gibt es Speicher, die mit Kondensatoren arbeiten. Man nennt sie dynamische Speicher. Da bedeutet ein geladener Kondensator 1 und ein ungeladener 0. Die Kondensatoren dieser Speicher verlieren aber nach etwa 2 bis 4 ms ihre Informationen. Daher müssen die Daten vor Ablauf dieser Zeit immer wieder neu aufgefrischt werden. Hilfsschaltungen für diesen Zweck sind dabei meist mit in das IC eingebaut, man nennt den Vorgang englisch „Refresh".

Allerdings muß der Refresh von außen angestoßen werden und es muß dafür meist auch eine Speicheradresse mitgegeben werden (zumindest von einer Spalte oder Reihe), deren Inhalt dann automatisch aufgefrischt wird. Diese dynamischen Speicher sind im allgemeinen mit einer viermal so hohen Kapazität verfügbar als die sogenannten statischen Speicher. Beim dynamischen Speicher ist nämlich nur ein Transistor zur Abfrage der Kondensatoren nötig, während bei den statischen Speichern, die mit Flipflops arbeiten, zu den zwei Flipflop-Transistoren nochmals zwei Transistoren kommen, die das Flipflop im Betrieb bedienen. Dynamische Speicher gibt es zur Zeit mit 1 048 576 Bit, (im Labor 4 194 304) auf einem Chip, während bei statischen Speichern derzeit bis zu 262 144 Speicherzellen auf einem Chip verfügbar sind.

Ein neuer Schritt beim Aufbau

Zum Abschluß wird die SBC2-Baugruppe weiter aufgebaut. Dazu werden die beiden Decoder 74LS138 in die zwei 16poligen Fassungen gesteckt (Wo ist Pin 1?). Der eine Decoder übernimmt die Aufgabe der Decodierung (Zuordnung einer Adresse zu einem Speicherbaustein) für die RAM-Bausteine, die später in die Fassungen 2 (IC8) und 3 (IC9) gesteckt werden und der andere für die EPROMs, die in die Fassungen 0 (IC6) und 1 (IC7) kommen. Es ist nämlich so, daß zum Beispiel die verwendeten statischen Speicherbausteine nicht volle 65 536 Byte fassen, sondern nur 2048. Deshalb besitzen sie auch nur 11 Adreßleitungen herausgeführt. Diese sind mit den Adreßleitungen A0 bis A10 der CPU verbunden. Um nun Speicher ICs „aneinanderreihen" zu

können, haben sie den Aktivierungseingang, der von einem externen Decoder bedient werden kann. Der hier verwendete Decoderbaustein könnte bis zu 8 Speicherbausteine bedienen.

Wir wollen die Decoder einmal testen. Dazu wird nochmal der Nichts-tu-Stecker benötigt. Er kommt in Fassung 0 (IC16). Nach dem Einschalten kann man nun am RAM-Decoder und am ROM-Decoder messen. Pin 15, 14, 13, 12, 11, 10, 9 und 7 sind die Ausgänge der Decoder. Dort müssen Pulse sichtbar werden. Wenn man den Prüfstift verwendet, so kann man dies auch deutlich erkennen. IC4 ist der EPROM-Decoder und IC5 der RAM-Decoder. Pin 15 des EPROM-Decoders zeigt etwas andere Signale als die restlichen Anschlüsse, das ist jedoch normal und hängt mit dem Aufbau des Z80 zusammen. Wenn man beide Decoder vergleicht, sieht man, daß die Signale recht ähnlich aussehen. Man kann hier einmal mit einem Zweikanal-Oszilloskop Vergleiche durchführen und wird feststellen, daß die Signale zueinander zeitlich versetzt sind. Die CPU spricht ja nacheinander alle Speicheradressen an (auch wenn sie nicht vorhanden sind). Seine Eingänge sind an den höherwertigen Adreßleitungen des Z80 befestigt. Wenn man mit dem Skop genauer hinsieht, so kann man feststellen, daß die Auswahl eines bestimmten Speicherbausteines eine bestimmte Zeit lang durch viele kleine Pulse wiederholt geschieht. Genau sind es 2048 beim RAM und 4096 beim EPROM. Aber bitte nicht versuchen, sie abzuzählen. Ausnahme bildet Pin 15 des ROM-Decoders, denn dort sind noch mehr Impulse zu finden als nur die Lese-Pulse für den Speicher. Und hier noch etwas zu Speicherorganisationen.

Unser RAM-Speicher hat eine Kapazität von 16384 Bit. Hier sind die Flip-Flops nicht in einer einfachen 128 × 128-Matrix angeordnet, sondern als 64 × 32 × 8-Matrix.

Das bedeutet, es sind acht Ebenen vorhanden. In diesen Ebenen werden jeweils bei Anlegen einer Adresse je genau ein Flipflop angewählt, so daß auf 8 Datenleitungen ein Byte zur Verfügung steht. Weiterhin fällt im Schaltbild vielleicht auf, daß es keine getrennten Daten-Ein- und Ausgänge gibt. Dies ist auch nicht unbedingt nötig, denn bei diesen Speichern wird immer nur entweder gelesen oder geschrieben, so daß man die Leitungen für Ein- und Ausgang gemeinsam benutzen kann. Wir haben also auch hier „bidirektionale" Leitungen, wie wir sie schon beim Z80 kennengelernt haben.

Zum Nachdenken

1. Was ist der Unterschied zwischen ROM und EPROM?
2. Warum kann man ein EPROM nicht als RAM verwenden und welche Nachteile hätte das?
3. Wieviel Adreßleitungen besitzt der Speicher 6116, den wir auf der SBC2-Baugruppe verwenden? Er besitzt eine Organisation von 2048 × 8 Speicherzellen, und 8 Datenleitungen sind direkt herausgeführt (Lösung siehe Schaltplan, durch Abzählen der Leitungen A0, A1, A2 . . .).

4.2 Die Z80-CPU voll ausgebaut

Wenn Sie die SBC2-Karte aufgebaut haben, dann werden Sie festgestellt haben, daß darin die Fähigkeiten des Z80 nicht vollständig ausgenutzt wurden. Vor allem waren die Speicheradressen beschränkt, die der CPU für ihre Arbeit zur Verfügung standen. Das liegt daran, daß diese Karte besonders preiswert sein sollte, damit jedermann zunächst einmal einen vollständigen Computer in Betrieb nehmen kann. Wenn jemand danach feststellt, daß die Sache mit den Computern wirklich Spaß macht, dann kann und muß er mit einem neuen Konzept professionell einsteigen,

```
     -5V  o   1   |       |       |
    +12V  o   2   |       |       |
    -12V  o   3   |       |       |
     +5V  o   4   |       |       |   S
     +5V  o   5   |       |       |
      0V  o   6   |       |       |   P
      0V  o   7   |       |       |
|     D0  o   8   |       |       |   E
|     D1  o   9   |       |       |
|     D2  o  10   |       |       |   I
|     D3  o  11   |       |       |
|     D4  o  12   |       |       |   C
|     D5  o  13   |       |       |
|     D6  o  14   |   S   |       |   H
|     D7  o  15   |       |       |
     -RD  o  16   |   B   |   I   |   E
     -WR  o  17   |       |       |
   -IORQ  o  18   |   C   |   O   |   R
   -MREQ  o  19   |       |       |
|     A0  o  20   |       |       |
|     A1  o  21   |   -   |   -   |   -
|     A2  o  22   |       |       |
|     A3  o  23   |       |       |
|     A4  o  24   |   B   |   B   |   B
|     A5  o  25   |       |       |
|     A6  o  26   |   U   |   U   |   U
|     A7  o  27   |       |       |
  -RESET  o  28   |   S   |   S   |   S
     -M1  o  29   |       |       |
     PHI  o  30   |       |       |
   -RFSH  o  31 ------------------|       |
    -INT  o  32                           |
   -WAIT  o  33 --------------------------|
|     A8  o  34
|     A9  o  35
|    A10  o  36
|    A11  o  37
|    A12  o  38
|    A13  o  39
|    A14  o  40
|    A15  o  41
  BANKEN  o  42
  -BUSRQ  o  43
  -BUSAK  o  44
      PI  o  45
      PO  o  46
    -NMI  o  47
     A16  o  48
     A17  o  49
     A18  o  50
     A19  o  51
      0V  o  52
      0V  o  53
 Reserve  0  54
```

Abb. 4.2.1 Der Bus ist logisch-physikalischer Träger des Systems

während jemand ohne Vorliebe für Computer nur wenig investiert hat und ohne Reue und mit vielen wertvollen Kenntnissen versehen, jetzt aussteigen kann.

Ein modulares Konzept

Betrachten Sie *Abb. 4.2.1*. Es zeigt ein Schema, das alle Z80-Signale und noch einiges mehr enthält. Es ist die logische Grundlage des NDR-Klein-Computersystems, die vor allem die nahezu beliebige Ausbaubarkeit des Gerätes garantiert. Im Grunde ist es nur eine Beschreibung eines Leitungspaketes, das auf einer Platine (wenn es die Normalausführung ist) sechs Steckplätze untereinander verbindet. Diese Platine haben Sie schon kennengelernt, es ist die Busplatine, die auch das physikalisch tragende Element in diesem Computersystem bildet. Die SBC2-Platine hat nur etwa die Hälfte dieser Leitungen ausgenutzt, von denen bis auf zwei Ausnahmen alle parallel von Sockel zu Sockel laufen.

Der Zweck eines solchen Bussystems ist einfach zu beschreiben. Da im Anwendungsfeld eines Computers immer wieder neue Anforderungen auftreten können, muß ein ordentlicher Computer möglichst einfach erweiterbar sein, damit er zum Beispiel mehr Speicher bekommen kann, oder etwa eine Platine, mit der er besonders schnell auf Signale von außen reagieren kann – oder anderes mehr. Dabei geht es immer darum, daß über bestimmte Kontrollsignale festgelegt werden muß, ob gelesen oder geschrieben werden soll, und dies von einer oder in eine bestimmte Adresse. Ein solcher Ablauf ist in seiner Reihenfolge so elementar, daß sich die verschiedenen Mikroprozessortypen mehr in der Bezeichnung der beteiligten Signale unterscheiden, als durch den physikalisch-logischen Ablauf dabei. Immer muß erst eine Adresse auf den Adreßleitungen ausgesandt werden und dann müssen entsprechend der gewünschten Aktion die Steuersignale bereitgestellt werden, wobei auch das Timing dabei weitgehend natürlich ist. Denn zum Beispiel kann ein Speicherbaustein erst einige Zeit nach seiner Anwahl Daten freigeben oder empfangen. Gleiches gilt auch für Peripheriebausteine. Das Bussystem des NDR-Computers ist nun so konstruiert, daß es besonders einfach ist und daß es die Peripherie und Speicherkarten nach Art des Z80 (mit einigen Erweiterungen für einen Ausbau mit modernen 8-/16-Bit-Prozessoren) bedienen kann. Der Z80-Prozessor ist heute einer der bewährtesten Computerchips, den die Industrie massenweise einsetzt. Die Signale anderer Prozessortypen lassen sich meist mit nur wenig Zusatzlogik an diesen Bus anpassen.

Die Vollausbau-CPU

Wenn man auf der SBC2-Platine ein Programm schreiben würde, das sehr hohe Adressen, etwa über A000h, ansprechen wollte, dann würde unter diesen Adressen wieder etwas gelesen, was auch schon unten im Adreßraum steht, denn die höherwertigen Adreßleitungen werden einfach nicht zur Speicheranwahl ausgewertet. Der Prozessor kann also in solchen Fällen Adreßbereiche doppelt sehen. Das darf natürlich keinesfalls in Systemen geschehen, wo jede Adresse auch mit Speicherzellen ausgerüstet ist. Deshalb muß eine neue CPU-Karte eingesetzt werden. Deren Schaltplan zeigt *Abb. 4.2.2*.

Im Grunde sind dort die Z80-Signale nur noch über geeignete Verstärker-ICs geleitet, um dann auf den Bus zu gelangen. Auf der Platine ist neben der CPU, der Reset-Logik und der Taktlogik nichts weiter vorgesehen. Das ergibt einen klaren Funktions-Block, eben die CPU. Die Taktlogik ist etwas raffinierter aufgebaut als bei der SBC2-Platine, weil an die Präzision und die Belastbarkeit der Signale jetzt erhöhte Anforderungen gestellt werden müssen. Ein Oszillator mit 8-MHz-Quarz erzeugt einen Takt, der von IC3 halbiert wird und dann über Verstärker sowohl an die CPU

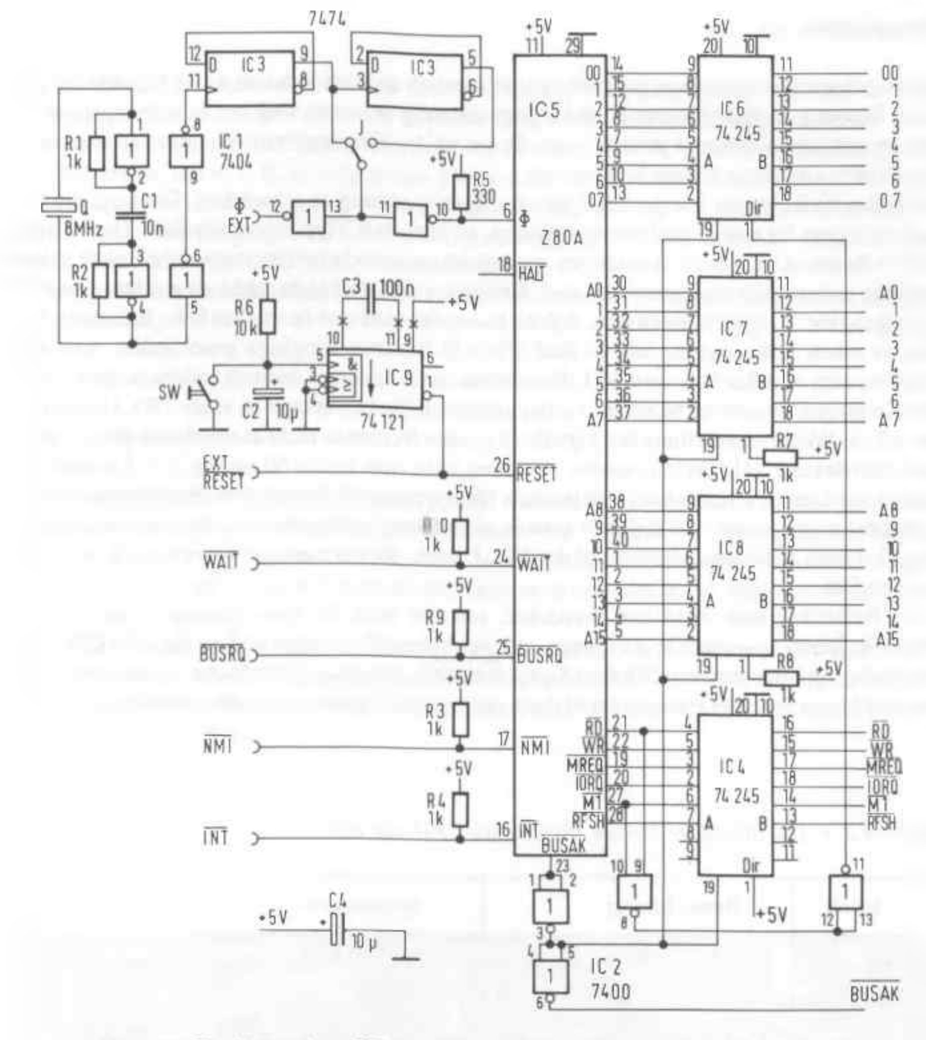


Abb. 4.2.2 Die Vollausbau-CPU mit Z80, eine in sich geschlossene Baugruppe

als auch an den Bus weitergegeben wird. Die Reset-Logik ist genauso wie bei der SBC2-Platine aufgebaut. Ein Monoflop (IC9) erzeugt bei Betätigung des Schalters SW einen präzisen Impuls, der die CPU zurücksetzt, ohne den von der CPU durchgeführten Refresh dynamischer Speicherbaugruppen zu lange zu blockieren. Wiederum ist das Signal an den Bus geführt, um auch andere Karten rücksetzen zu können.

Die eigentliche Bussteuerlogik wird von den ICs 2, 4, 6, 7, 8 gebildet.

Bus-Puffer sind notwendig

Solche bidirektionalen Bus-Puffer, wie die ICs 4, 6, 7, 8 im Slang genannt werden, findet man oft in größeren Mikroprozessorsystemen. Sie werden benötigt, weil viele der hochintegrierten ICs

relativ „schwache“ Ausgänge besitzen. Es ist nämlich so, daß jedes an einer Signalleitung mit einem Eingang angeschlossene IC diese Signalleitung belastet, weil bei Schaltvorgängen und auch im statischen Zustand je nach Lage Strom in das Eingangsbein des angeschlossenen ICs hinein- oder aus diesem Bein herausfließt.

Und das leider immer mit der Tendenz, die Signalspannung zu schwächen. Die Last, die solch ein IC-Eingang für eine Signalleitung bedeutet, ist je nach IC-Typ unterschiedlich. Deshalb kann ein Z80-Baustein mit seinen Ausgängen nur Signalleitungen betreiben, an welchen nicht zu viele Eingänge anderer ICs angeschlossen sind. Beispielsweise darf nicht mehr als ein Standard-TTL-IC-Eingangsbein angeschlossen sein. Solche Bausteine sind eine besonders hohe Belastung. Man kann an einen Z80-Ausgang bis zu fünf TTL-LS-Baustein-Eingänge anschließen, weil diese Bausteine sich von den Standard-TTL-Bausteinen unter anderem dadurch unterscheiden, daß sie Signalleitungen nur wenig belasten. Leider sind auch fünf Eingänge an einer Z80-Ausgangsleitung oft zur Weiterverarbeitung der Signale in großen Systemen nicht ausreichend. Setzt man zur Signalverstärkung 74LS245-Bausteine ein, dann kann man bis zu 60 andere TTL-LS-Bausteine darüber mit Signalen versorgen. Das ist auch für Systeme mit Bussteckplätzen (wie beim NDR-Computer) ausreichend, bei welchen man ja nicht immer vorhersehen kann, wie viele ICs ein Signal belasten. Übrigens, die Anzahl der TTL-Lasten, die ein Baustein treiben kann, nennt man sein Fan Out.

Die Bustreiber sind nicht nur Verstärker, sondern auch Tri-State-Elemente, das heißt, sie können in ihrem Inneren alle Ausgänge auf „hochohmig“ schalten und so die Z80-CPU (IC5) vollständig vom Bus trennen. Mit dem Signal $\overline{\text{BUSRQ}}$, das von außen gegeben werden muß, kann das erreicht werden. Der Prozessor gibt dann ein Signal $\overline{\text{BUSAK}}$ aus, wenn er den Bus nicht mehr

Tabelle 4.2.1 Die Stückliste für die Vollausbau-CPU mit Z80

Stück	Bezeichnung	Bauelement
1	IC1	74 LS 04
1	I2	74 LS 00
1	IC3	74 LS 74
4	IC4, IC6, IC7, IC8	74 LS 245
1	IC5	Z80-A-CPU
1	I9	74 [illegible] 121
4	SO14	14polige IC-Fassung
4	SO20	20polige IC-Fassung
1	SO40	40polige IC-Fassung
8	R1, 2, 3, 4, 7, 8, 9, 10	1 kΩ
1	R5	330 Ω
1	R6	100 kΩ
1	C1	10 nF
2	C2, C4	10 μF
1	C3	100 nF
1	T1	Taster
1	Q1	Quarz 8 MHz
1	St1 (Stecker)	18- und 36polige Steckerleiste
1	Platine mit Lötstoplack	

benötigt. Die LS245-Bausteine werden über den Tri-State-Eingang (Pin 19) in den Tri-State-Zustand geschaltet. Es liegen dann nur noch Reset und der Takt direkt auf dem Bus.

Damit ist es mit speziellen Bausteinen, z. B. sogenannten DMA-Controllern, möglich, auf den Bus zuzugreifen, um so z. B. schnell an den Speicher heranzukommen. Der Eingang $\overline{\text{WAIT}}$ dient zum Anhalten der CPU, wenn z. B. eine Peripherieeinheit zu langsam ist. Dann muß diese bei einem Zugriff den Eingang $\overline{\text{WAIT}}$ auf 0 V legen, so lange, bis die Daten korrekt verarbeitet wurden. Dann kann die CPU wieder weiterarbeiten.

Interrupts können mit den Eingängen $\overline{\text{INT}}$ und $\overline{\text{NMI}}$ verarbeitet werden. Doch das ist nur etwas für Spezialisten.

Aufbau und Test

Der Aufbau der CPU-Platine ist besonders einfach, weil doch recht wenig ICs auf der Platine sind. Wie immer setze man zuerst die passiven Bauelemente ein, dann die IC-Sockel und erst dann die ICs selbst. Auf die Lage von Pin 1 achten! *Tabelle 4.2.1* zeigt die Stückliste, *Abb. 4.2.3* zeigt die Lötseite der Leiterplatte, *Abb. 4.2.4* die Bestückungsseite und *Abb. 4.2.5* zeigt den Bestückungsplan.

Die Vollausbau-CPU läßt sich zunächst nur grob vortesten, dazu wird der Takt an Pin 6 der CPU gemessen, er muß 4 MHz (normalerweise) betragen. An Pin 26 muß bei Betätigen der Reset-Taste ein kurzer Reset-Puls erscheinen. Die restliche Funktion kann man nur zusammen mit einer Speicherbaugruppe testen.

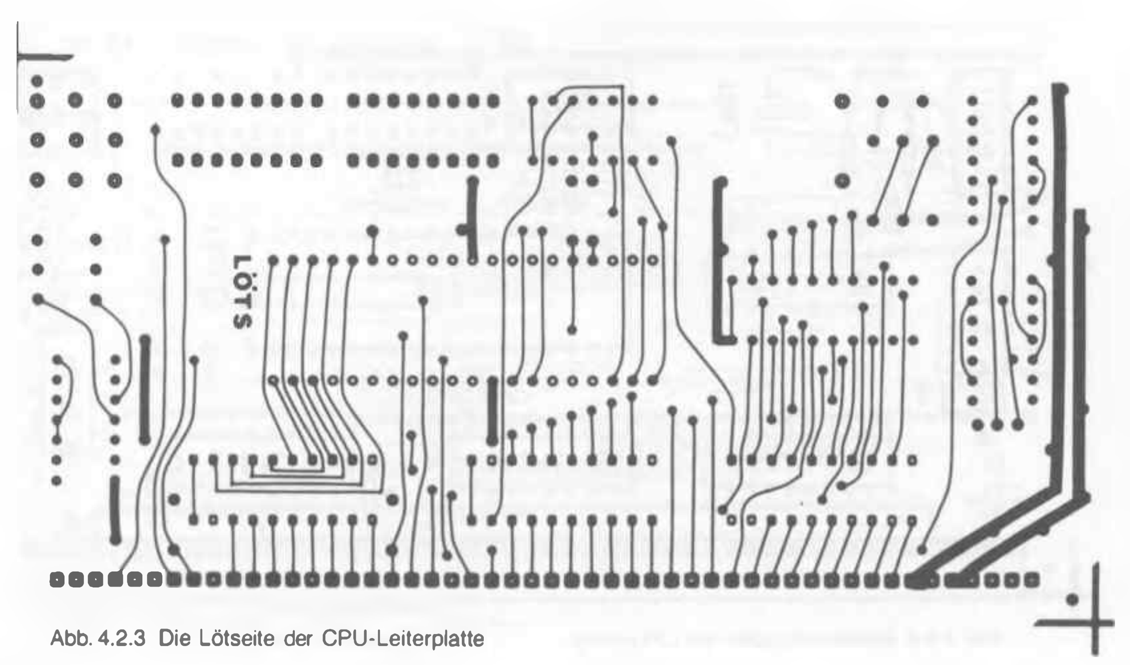


Abb. 4.2.3 Die Lötseite der CPU-Leiterplatte

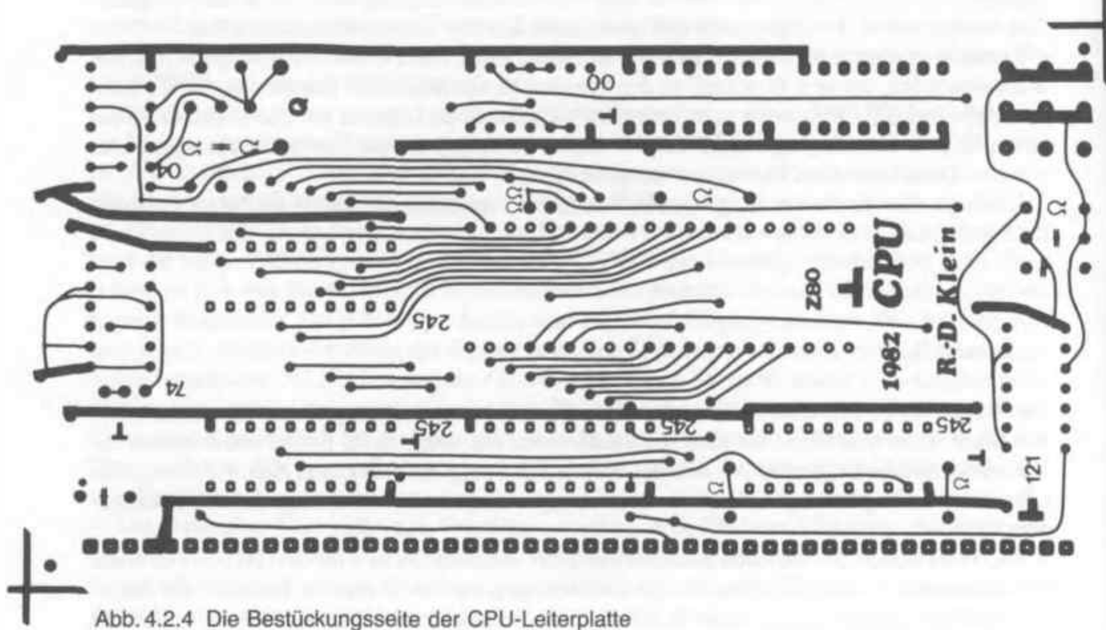


Abb. 4.2.4 Die Bestückungsseite der CPU-Leiterplatte

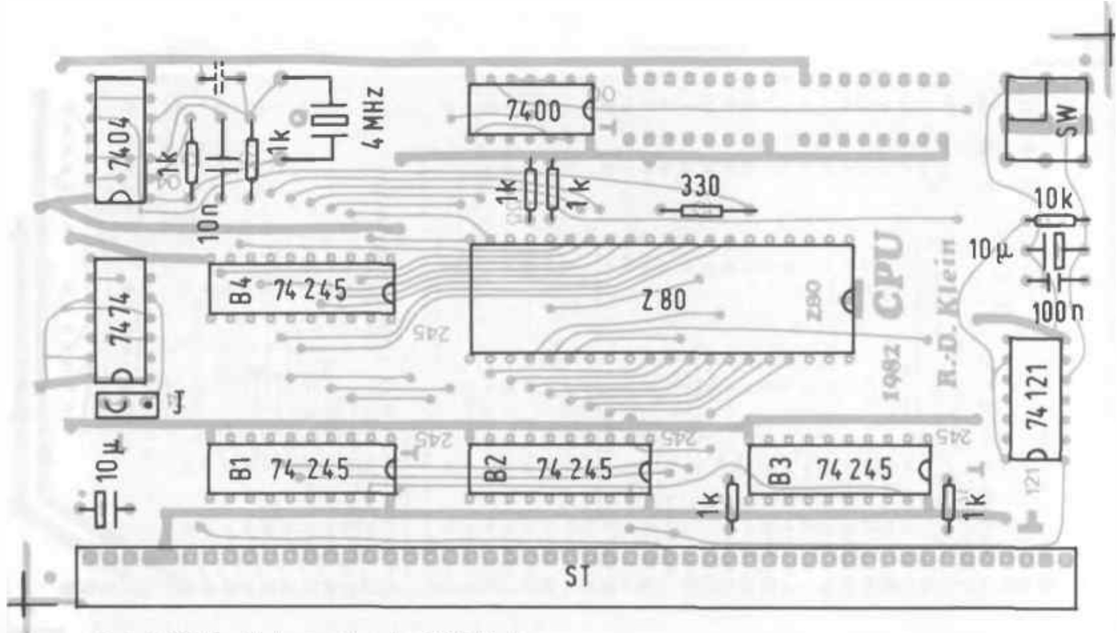


Abb. 4.2.5 Bestückungsplan der CPU-Karte

Zusammenfassung „die Z80-CPU"

Abb. 4.2.6 zeigt die Verbindung eines Mikroprozessors mit dem Speicher (ROM und/oder RAM). Als CPU werden wir den Prozessor Z80 von Zilog verwenden. Der Anschluß einer CPU an den Speicher erfolgt ähnlich zu dem, was wir bei dem vorigen Abschnitt schon getan haben.

Die CPU besitzt einen Adreßbus, auf dem z. B. die Programmmzähleradresse liegen kann. Dann gibt es einen Datenbus, über den der gesamte Informationsaustausch durchgeführt wird. Für die Steuersignale werden beim Z80 für den Speicher drei Leitungen verwendet. Die Leitung -RD (oder $\overline{RD}$) gibt an, wann ein Lese-Zugriff erfolgt. Dann liegt der Pegel auf 0. Bei -WR wird der Schreibzugriff durch ein 0-Signal angegeben. -RD und -WR sind daher niemals zur gleichen Zeit auf 0. Der Ausgang -MREQ schließlich zeigt an, ob ein Speicherzugriff gewünscht wird. Im Gegensatz dazu gibt es beim Z80 auch noch einen I/O-Zugriff, auf den wir später noch zurückkommen. Abb. 4.2.7 zeigt einen Lese-Zugriff der CPU, und Abb. 4.2.8 zeigt einen Schreibzugriff.

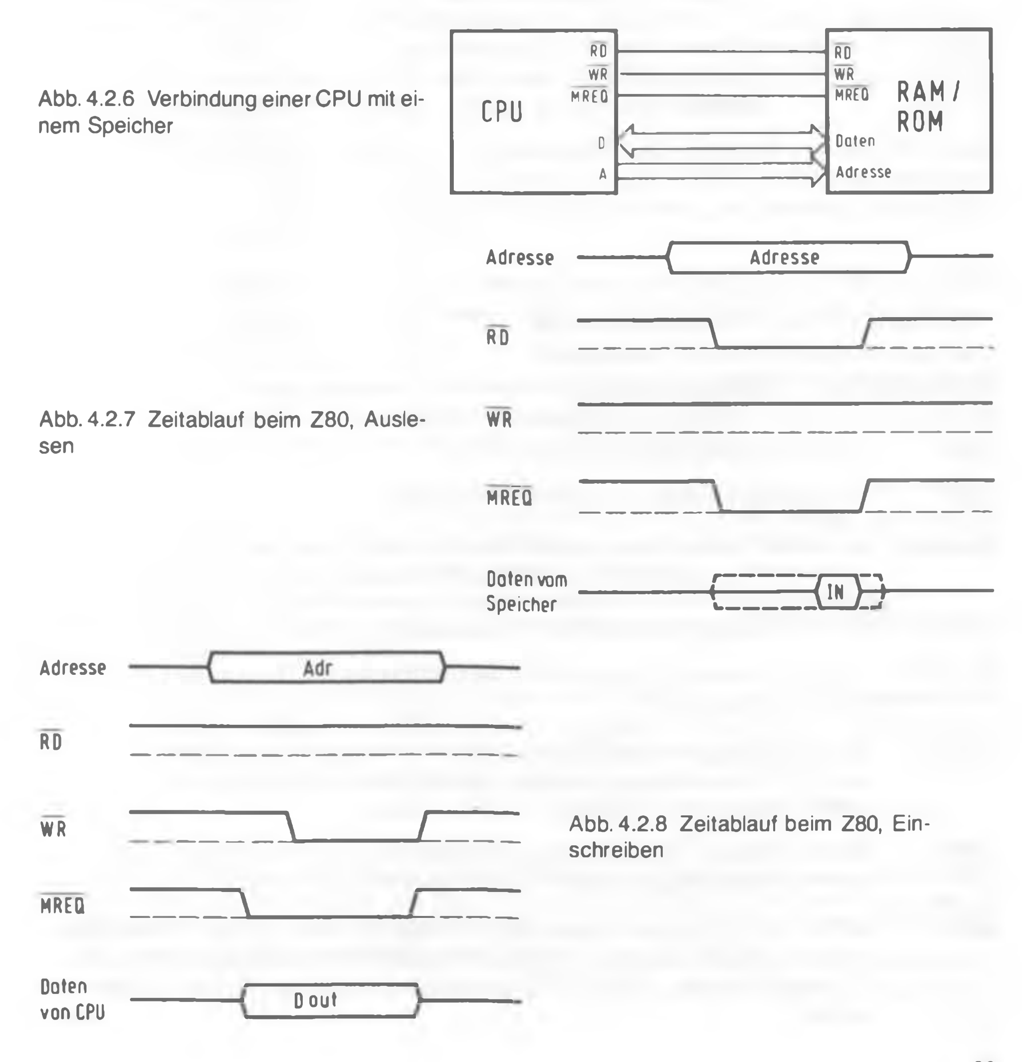


Abb. 4.2.6 Verbindung einer CPU mit einem Speicher

Abb. 4.2.7 Zeitablauf beim Z80, Auslesen

Abb. 4.2.8 Zeitablauf beim Z80, Einschreiben

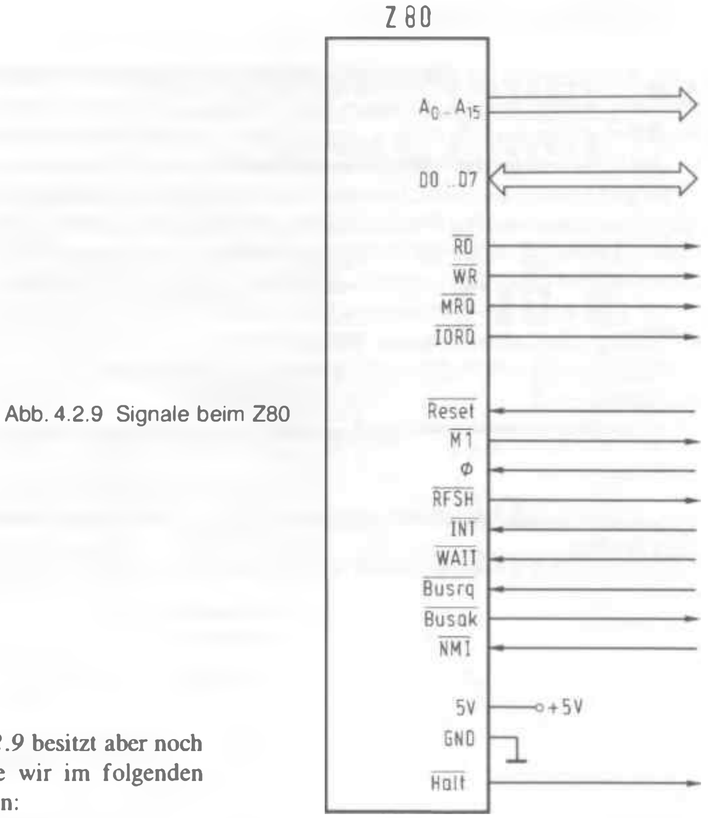


Abb. 4.2.9 Signale beim Z80

Die Z80-CPU *Abb. 4.2.9* besitzt aber noch viele andere Signale, die wir im folgenden einzeln durchgehen wollen:

GND — Masseanschluß der CPU

+ 5 V — Versorgungsspannung, die stabilisiert sein muß.

A0 . . A15 — Tri-State-Adreßbus. Bei Speicherzugriffen können damit bis zu 65536 (2 hoch 16) Speicherzellen angesprochen werden. Bei I/O-Zugriffen liegt auf den unteren 8 Leitungen (A0 . . A7) die Adresse eines Peripherie-Bausteins und beim Refresh-Zyklus (für dynamische Speicher) auf den unteren 7 Leitungen die Refresh-Adresse.

D0 . . D7 — Tri-State Datenbus. Der Bus wird für die Übertragung von Daten an die CPU und von der CPU verwendet.

-M1 — Maschine Cycle One. Ein 0-Signal auf dieser Leitung zeigt an, daß die CPU gerade einen Befehlscode holt. -M1 tritt auch zusammen mit -IORQ auf, wenn ein Interrupt quittiert wird (siehe Interrupts im Software-Kapitel).

-MREQ — Memory Request. Dieser Tri-State-Ausgang zeigt bei einem 0-Signal an, daß der Adreßbus eine gültige Adresse für einen Speicherzugriff enthält.

-IORQ — Input/Output Request. Durch diesen Tri-State Ausgang wird bei einem 0-Signal angezeigt, daß auf den unteren 8 Bits des Adreßbusses eine gültige Adresse für einen I/O-Zugriff vorliegt. Tritt -IORQ gleichzeitig mit -M1 auf, so wird ein Interrupt quittiert.

-RD Read Access. Mit dem Tri-State-Ausgang wird der Wunsch eines Lesezugriffs durch ein 0-Signal angegeben, dabei entscheidet sich durch das Signal -IORQ oder -MREQ, ob von einer Peripherie oder Speichereinheit gelesen werden soll.

-WR Write Access. Durch diesen Tri-State-Ausgang wird bei einem 0-Signal ein Schreib-Zugriff angekündigt.

-RFSH Refresh. Ein 0-Signal gibt einen Refresh-Cyclus an. Dann liegt auf den unteren 7 Bits des Adreß-Busses eine Adresse an, die durch einen im Z80 befindlichen Zähler bestimmt ist. Diese Adresse kann bei dynamischen Speichern zum Wiederauffrischen der internen Zellen verwendet werden.

-HALT Halt State. Liegt der Ausgang auf einem 0-Signal, dann wurde von der CPU zuvor ein HALT-Befehl ausgeführt. Bei dem Halt werden NOPs zur Aufrechterhaltung des Refresh ausgeführt. Aus dem Halt-Zustand kann man nur durch Reset oder einen freigegebenen Interrupt herauskommen.

-WAIT Wait. Bei diesem Eingang kann durch ein 0-Signal der CPU gesagt werden, daß ein Speicher oder Peripheriegerät noch nicht bereit für einen Datenaustausch ist. Damit können auch langsame Peripherie oder Speichergeräte an die CPU angeschlossen werden. Ein Refresh wird in dieser Zeit nicht durchgeführt.

-INT Interrupt Request. Durch ein 0-Signal kann ein Interrupt gegeben werden. Das Signal wird am Ende eines Instruktions-Zyklus akzeptiert, falls der Interrupt freigegeben wurde und -BUSRQ nicht aktiv ist. Wurde der Interrupt angenommen, so wird dies durch -IORQ und -M1 bestätigt.

-NMI Non Maskable Interrupt. Der Eingang reagiert auf die negative Flanke und wird immer angenommen. Nach einem -NMI-Signal wird die Adresse 66H angesprungen. -BUSRQ darf nicht vorliegen.

-RESET Reset. Damit wird die CPU in den Grundzustand gesetzt. Der Programmzähler wird auf 0 gesetzt, und Interrupts werden gesperrt. Das Register I wird auf 0 gesetzt und ebenfalls das Register R. Der Interrupt wird auf Mode 0 gesetzt.

-BUSRQ Bus Request. Durch diesen Eingang wird bei einem 0-Signal der Zugriff auf den CPU-Bus verlangt. Damit kann von einem externen Gerät auf Speicher oder IO zugegriffen werden, ohne daß die CPU daran beteiligt wird. Alle Tri-State-Ausgänge der CPU werden in den offenen Zustand überführt.

-BUSAK Bus Acknowledge. Die CPU gibt durch ein 0-Signal an, daß sie den Bus für externe Geräte freigegeben hat.

PHI Phi. Der Takt der CPU. Bei der Standard-CPU Z80 sind maximal 2 MHz erlaubt; bei Z80A 4 MHz und bei Z80B 6 MHz. Der Eingang muß einen Pull-up-Widerstand von 330 Ohm erhalten, um von normalen TTL-Gattern angesteuert werden zu können; ansonsten können Störungen im Ablauf auftreten.

Die an den Anschlüssen vorliegenden Signale wollen wir nun im folgenden näher besprechen.

Abb. 4.2.10 zeigt die Relation des Taktes zu dem Befehlsablauf. Als Beispiel sei hier ein Befehl mit einem Lese- und nachfolgenden Speicherschreibzugriff gezeigt. Im M1-Cyclus wird der Befehlscode geholt. Dabei werden 4 Taktzyklen verbraucht. Ein Lese- oder Schreibzugriff benötigt nur 3 Taktzyklen. Alle zu einem Befehl gehörenden Zyklen nennt man Instruction-Cycles.

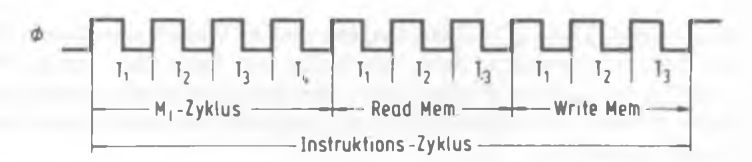


Abb. 4.2.10 Zeitablauf eines Instruktionszyklus

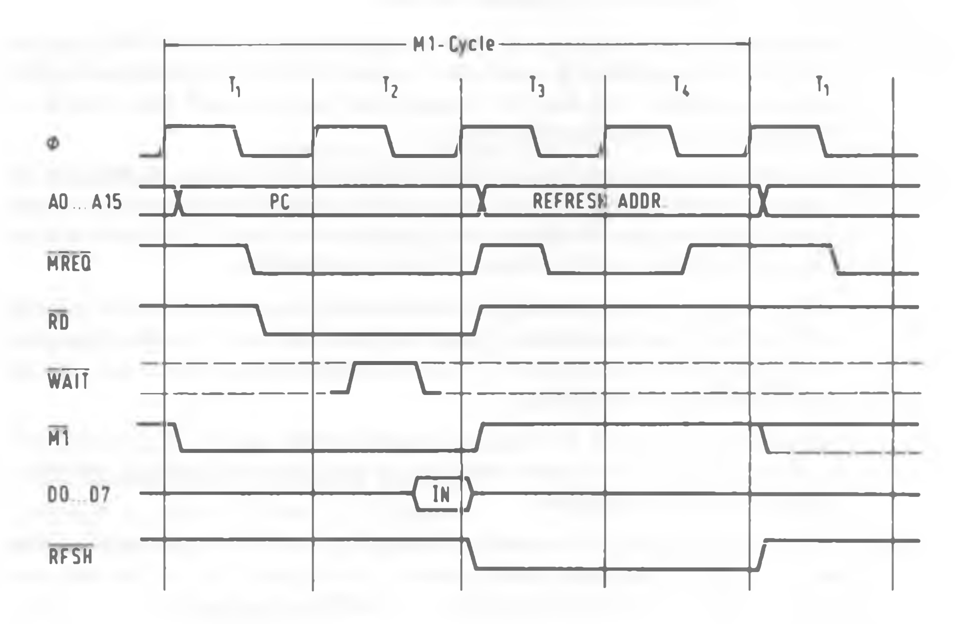


Abb. 4.2.11 M1-Zyklus

Abb. 4.2.11 zeigt nochmals genauer den Vorgang bei einem Instruktions-Zyklus. Bei den beiden Takt-Zyklen liegt auf dem Adreßbus die Programmzähleradresse an. Mit dem Signal -MREQ wird die Gültigkeit des Adreßbusses angegeben. Das Signal -M1 kennzeichnet eine Befehlshol-Phase. -MREQ bestimmt, daß ein Zugriff auf den Speicher erfolgen soll, und das Signal -RD gibt an, daß vom Speicher gelesen werden soll. Im zweiten Teil des Befehls-Zyklus wird eine Refresh-Adresse für dynamische Speicher ausgegeben.

Wird der Prozessor an Speicher angeschlossen, die zu langsam für einen normalen Zugriff sind, so kann die Zugriffszeit durch Einfügen eines sogenannten Wait-Zyklus verlängert werden. *Abb. 4.2.12* zeigt das Timing-Diagramm dazu. Der Eingang -WAIT wird bei der fallenden Flanke des Taktes bei einem Zugriff abgetastet. Ist die Leitung auf einem 0-Pegel, so wird ein Wait-Zyklus eingefügt. Ist die Leitung -WAIT beim nächsten Zugriff erneut auf 0, so wird wieder ein Wait-Zyklus eingefügt, bis die Leitung bei der fallenden Flanke des Takt-Signals einmal auf 1 war. Danach erfolgt der Zugriff, hier wird das Datenwort eingelesen. Der M1-Zyklus ist besonders zeitkritisch, weshalb es oftmals bei langsamen Speichern genügt, nur dann ein Wait-Signal zu liefern. Der Zeitverlust für die CPU ist dabei prozentual gesehen minimal.

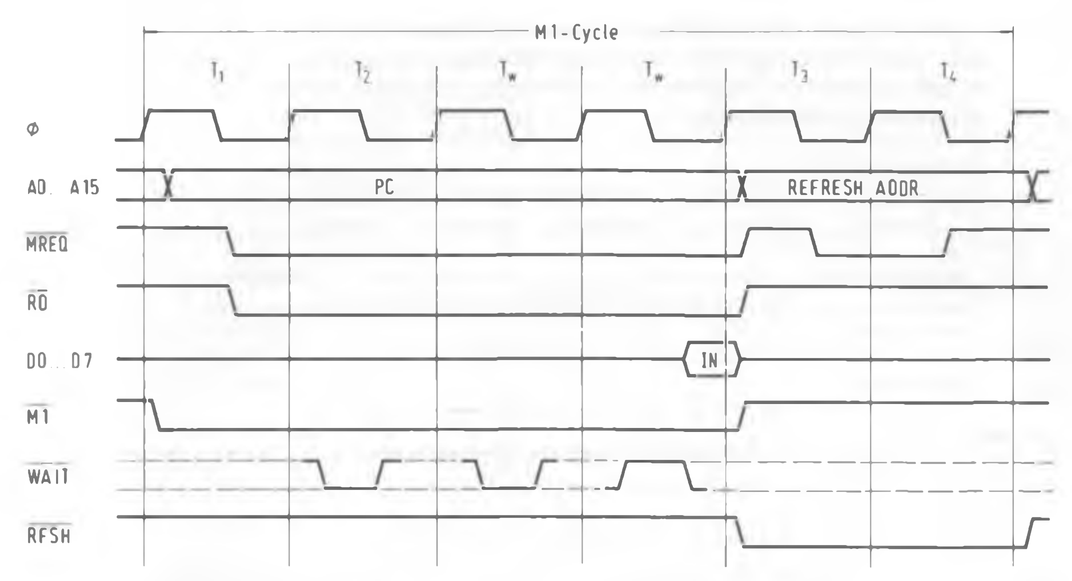


Abb. 4.2.12 M1-Zyklus mit Wait-Zyklen

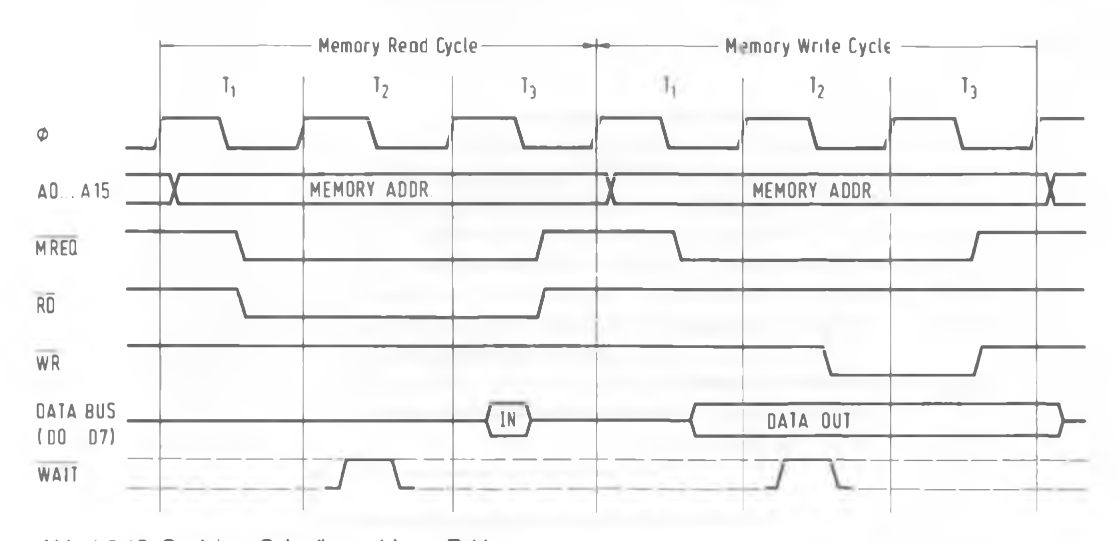


Abb. 4.2.13 Speicher, Schreib- und Lese-Zyklus

Abb. 4.2.13 zeigt den zeitlichen Ablauf bei einem Speicher Lese- und/oder Schreibzugriff. Die Ankündigung eines Lesezugriffs erfolgt mit den Signalen -MREQ und -RD. Eingelesen werden die Daten am Ende dieser Signale. Ein Schreibzugriff erfolgt mit den Signalen -MREQ und dem etwas später folgenden WR-Signal. Ebenso wie beim Instruktions-Zyklus kann auch hier durch Verwendung eines -WAIT-Eingangs ein Anschluß an langsame Speicher erreicht werden. *Abb. 4.2.14* zeigt die Situation.

Der Z80 kann neben den Speichern auch noch Peripherie adressieren. Als Adreßraum stehen bei einem Speicherzugriff 64 KByte zur Verfügung, wohingegen bei einem Zugriff auf eine Peripherie-Einheit nur 256 Adressen zur Verfügung stehen; der Z80 verwendet zur Adressierung die unteren 8 Adreßleitungen.

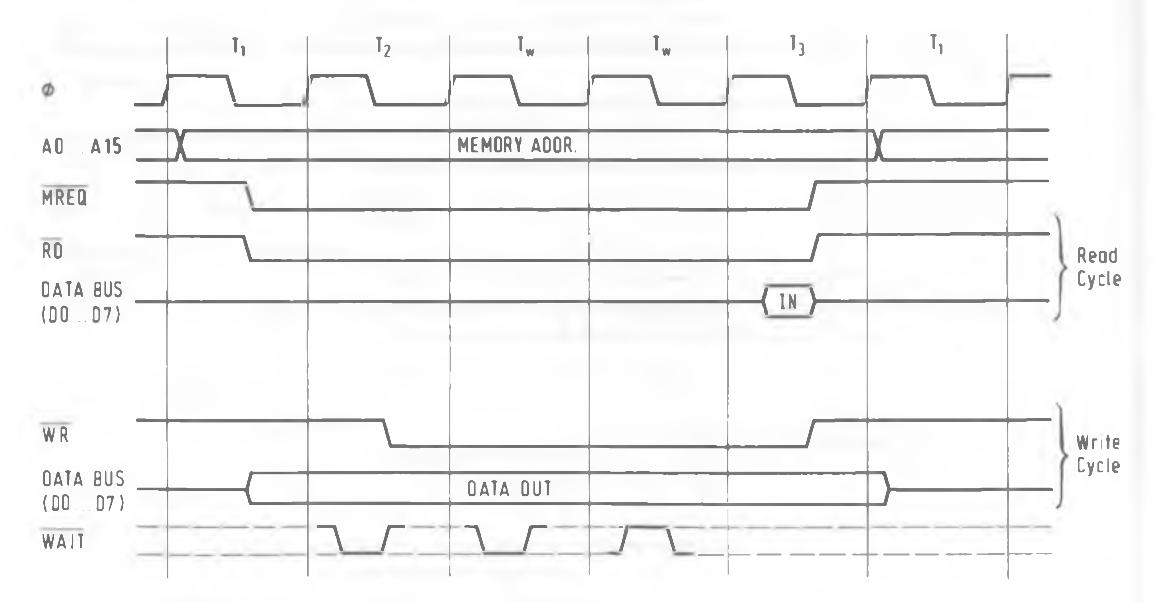


Abb. 4.2.14 Speicherzugriffe mit Wait-Zyklen

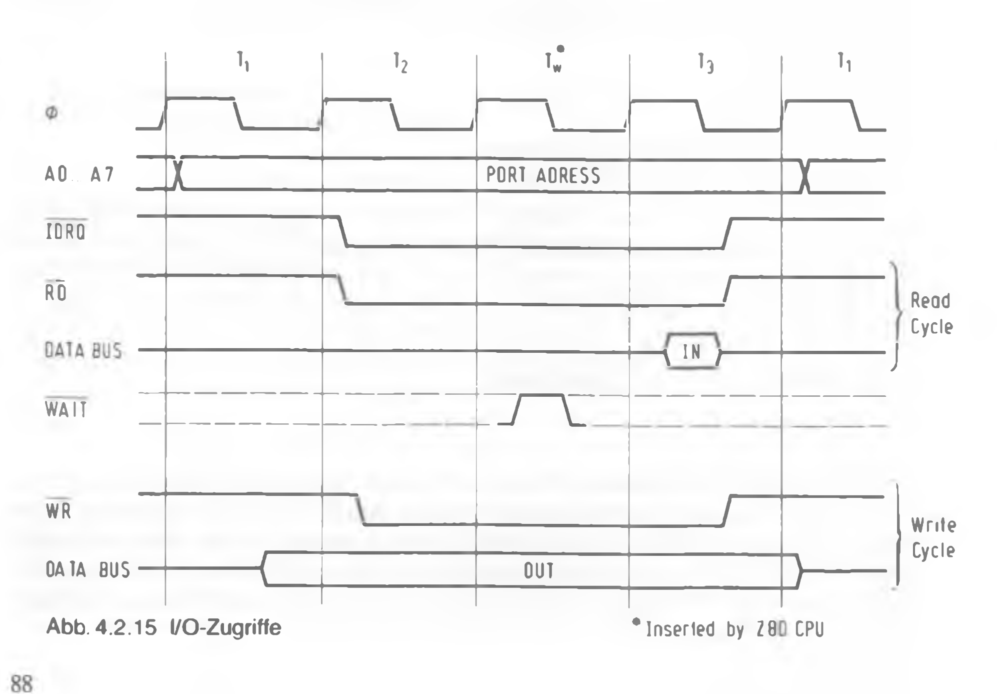


Abb. 4.2.15 I/O-Zugriffe

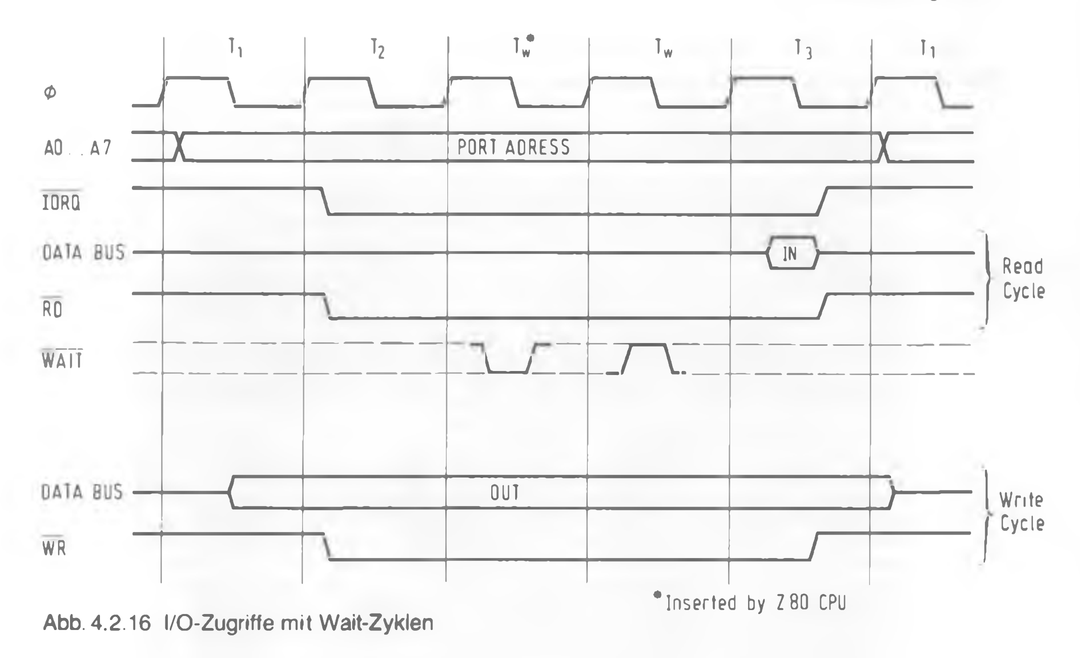


Abb. 4.2.16 I/O-Zugriffe mit Wait-Zyklen

Abb. 4.2.15 zeigt das Timing für einen Peripheriezugriff. Es läuft im Prinzip genauso wie bei einem Speicherzugriff ab, nur daß anstelle des Signals -MREQ die Leitung -IORQ aktiviert wird. Außerdem wird von der CPU automatisch ein Wait-Zyklus eingefügt, um es auch langsamen Peripherie-Bausteinen zu ermöglichen, ohne Zusatzschaltung Daten mit der CPU auszutauschen. Sollte dies dennoch nicht ausreichen, so ist es natürlich auch möglich, den Zugriff noch weiter zu verlängern, wie *Abb. 4.2.16* dargestellt ist. Dabei ist aber zu beachten, daß ein Refresh während der Zugriffszeit nicht durchgeführt wird und daher bei Verwendung von dynamischen Speichern eine maximale Zeit für einen Wait-Zyklus vorgegeben ist.

Damit zunächst genug vom Timing Z80. Auf die anderen Möglichkeiten soll an dieser Stelle nicht eingegangen werden, da wir sie für das weitere Verständnis nicht benötigen.

Der Bus

Damit wir unsere Baugruppe später miteinander verbinden können, brauchen wir eine Grundplatte, Bus-Baugruppe genannt.

Diese Bus-Platte verbindet im Prinzip alle Leitungen, die schon in *Abb. 4.2.1* gezeigt wurden, miteinander. Eine Ausnahme gibt es, die Signale PI und PO sind nicht durchgehend verdrahtet, sondern PO führt immer zum nächsten PI. Diese beiden Signale sind für eine sogenannte Daisychain vorgesehen und werden jedoch von allen bisherigen Baugruppen des NDR-Klein-Computers nicht verwendet.

Die Bus-Baugruppe ist im Buch im Europaformat (100 mm × 160 mm) abgedruckt. Man kann den Bus jedoch auch länger aufbauen, im Handel gibt es entsprechende Karten.

Abb. 4.2.17 zeigt die Lötseite der Bus-Baugruppe. *Abb. 4.2.18* zeigt die Bestückungsseite und *Abb. 4.2.19* zeigt einen kompletten Aufbau.

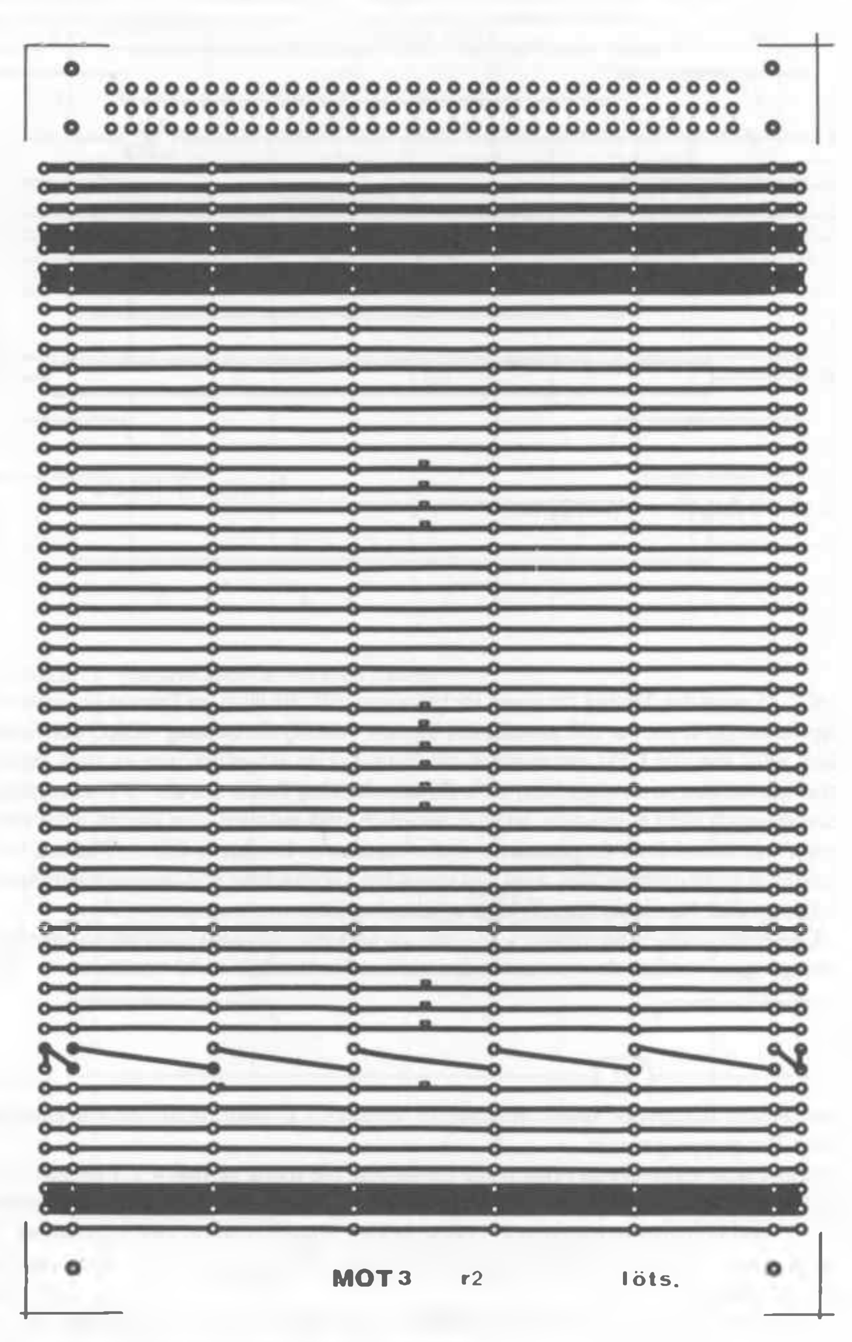


Abb. 4.2.17 Lötseite der Bus-Baugruppe

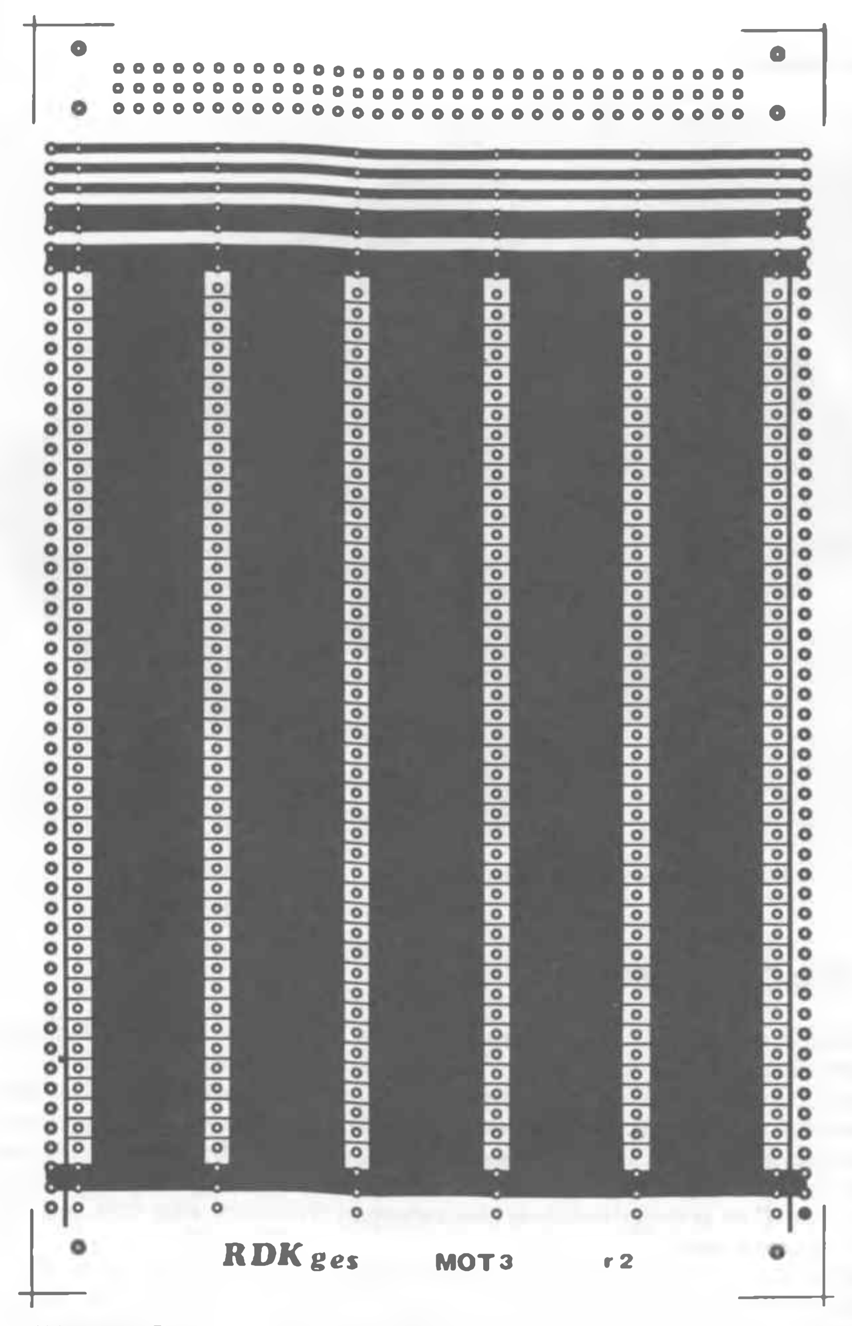


Abb. 4.2.18 Bestückungsseite der Bus-Baugruppe

Diese Karte ist ebenfalls zweiseitig und die zweite Seite ist besonders wichtig. Sie verbindet alle Masse-Leitungen miteinander. Die Masse-Leitungen sind recht großflächig ausgelegt, damit wird der Computer später störunanfällig.

Man sollte sich vor einer gefädelten, selbstgebauten Bus-Baugruppe hüten, das führt nur zu unnötigen Fehlern.

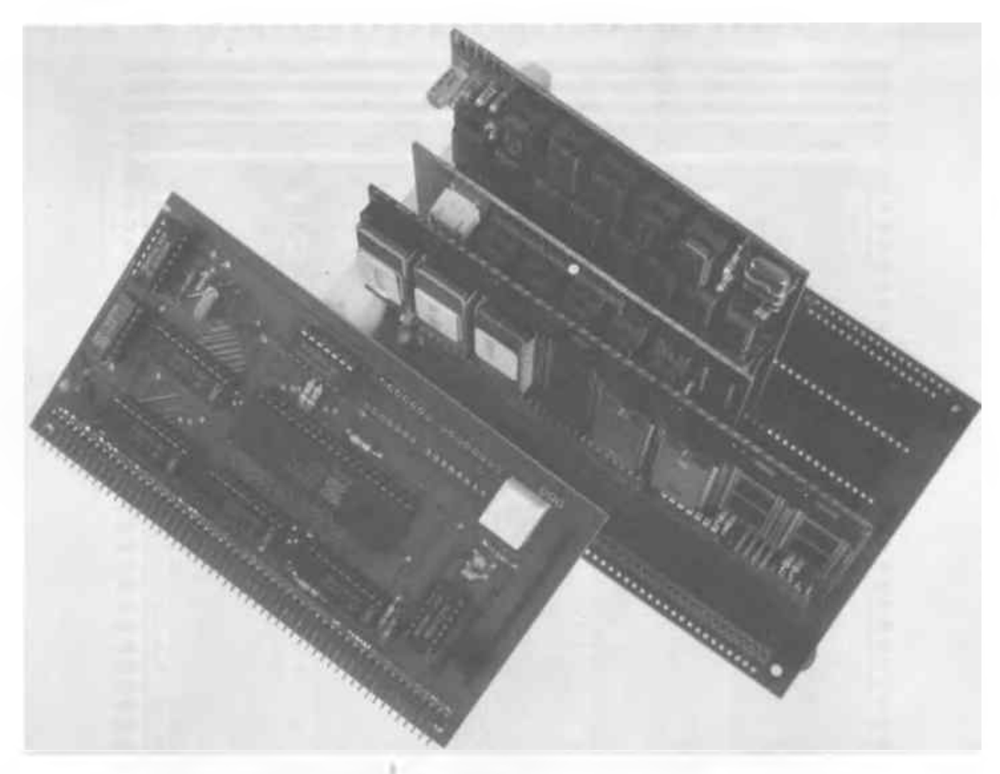


Abb. 4.2.19 So sieht die Busbaugruppe fertig aus

4.3 Die CPU 64180

Dieser Abschnitt ist für die Fortgeschrittenen gedacht, die gerne die modernste Technik haben möchten. Daher wird der Abschnitt auch entsprechend kurz ausfallen.

Die CPU mit der Typenbezeichnung HD64180 ist eine Weiterentwicklung der Z80-CPU. Sie besitzt einige neue Befehle, sowie zahlreiche interne Peripherieeinheiten. Eine Speicherverwaltung erlaubt es ihr bis zu 1/2 MByte (neuere Versionen sogar 1 MByte) anzusprechen. Ferner besitzt sie zwei serielle Schnittstellen, eine DMA-Einheit, einen Interrupt-Controller sowie Timer. Das IC ist in einem 64poligen Shrink-Gehäuse (Pinabstand nicht 2.54 mm, sondern 1.75 mm) untergebracht.

Abb. 4.3.1 zeigt die komplette Schaltung der Baugruppe. Die Baugruppe ist beim Bausatz-Hersteller erhältlich (siehe Bezugsquellenverzeichnis). Hier nur kurz ein paar Besonderheiten.

Wenn man den Bustreiber für die Adressen A16 . . A19 wegläßt, so ist die Baugruppe voll zur Z80-CPU kompatibel, und es laufen alle Programme unverändert. In dem IC muß man die MMU programmieren, um den zusätzlichen Adreßraum ansprechen zu können. Damit kann man sich aber z. B. die Baugruppe BANK/BOOT einsparen. Ferner werden Wartezyklen für IO und Speicher getrennt erzeugt und sind programmierbar (zwischen 0 und 3). Nach dem RESET wird automatisch die langsamste Rate eingestellt.

Die CPU arbeitet mit einem 8-Bit-Refresh, der sich aber auch abstellen läßt. Die NDR-Baugruppen werten das Signal normalerweise nicht zum Refresh aus. Mit der Brücke J2 kann man

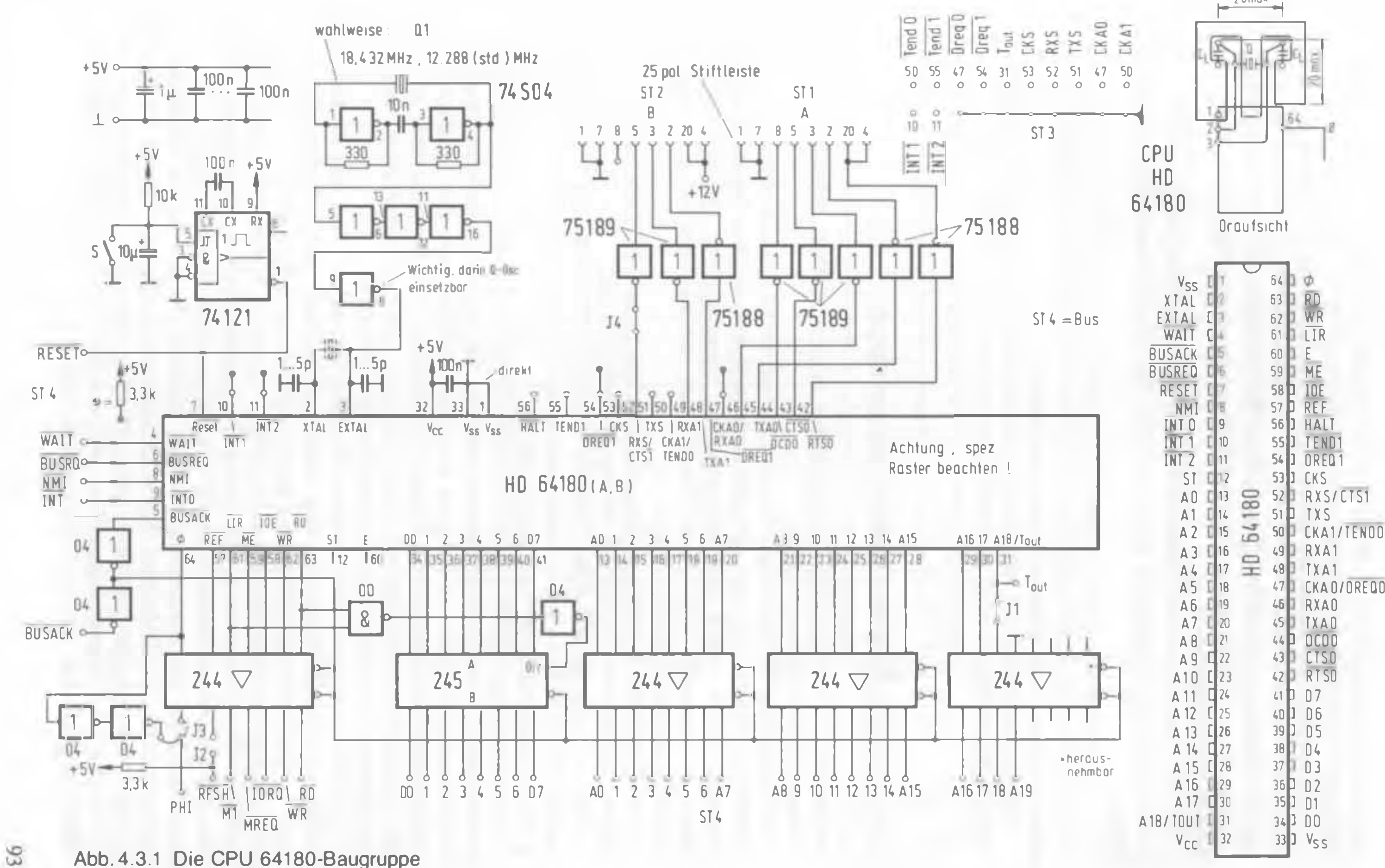


Abb. 4.3.1 Die CPU 64180-Baugruppe

das Signal vom Bus entfernen. Durch das 8te Bit kann es bei ein paar älteren Baugruppen zu Störungen kommen, daher also lieber die Brücke J2 offenlassen.

Mit J3 kann man den Takt auf den Bus legen. Wenn die Brücke mit den Invertern verbunden ist, so liegt der Takt dauernd an. Will man aber mit einem -BUSRQ-Signal auch den Takt trennen, so muß die Brücke so eingestellt werden, daß der Takt vom Bustreiber 74LS244 kommt.

Einige interessante Signale des HD64180 sind auf eine extra Steckerleiste gelegt, von wo aus man sie verwenden kann.

Die CPU besitzt einen internen Oszillator, hier wurde aber zur Sicherheit auch eine kleine Oszillatorschaltung vorgesehen, da der interne Oszillator bei den ersten Mustern noch Probleme machte.

Als Quarzfrequenz empfiehlt sich der Wert 12.288 MHz bei der B-Version, was einem 6 MHz Takt beim Z80 entspricht. Man kann aber auch einen 18.432 MHz-Quarz verwenden, wenn man einen schnellen 64180 bekommt (oder selbst selektiert).

Wer sich mehr für die CPU interessiert, sollte sich von der Firma Hitachi (oder Händlern) das ausführliche Handbuch besorgen, in dem auch die neuen Befehle erklärt sind.

4.4 Eine 64-KByte-Speicherbaugruppe

Da die beiden Vollausbau-CPUs keinen eigenen Speicher besitzen, müssen sie durch einen externen Speicher erweitert werden.

Die Baugruppe ROA64 ermöglicht es, den Speicher auf 64 KByte auszubauen. Mit weiteren ROA64-Baugruppen kann man den Speicherbereich noch darüber hinaus bis zu 1 MByte erweitern, benötigt jedoch noch eine zusätzliche Steuerbaugruppe (BANK/BOOT), die im Kapitel 4.5 vorgestellt wird. Beim HD64180 kann man allerdings bis zu 1/2 MByte direkt ansteuern, mit dem Z80 nur 64 KByte.

In die Baugruppe ROA64 kann man sowohl RAMs als auch EPROMs einstecken. Das ist möglich, da die 8K × 8-Bausteine miteinander kompatibel sind.

Abb. 4.4.1 zeigt den Schaltplan. 8 Bausteine vom Typ HM 6264 (oder kompatible RAMs), und/oder 2764 EPROMs sind vorgesehen. Jeder dieser Bausteine besitzt 8 KByte Speicherkapazität. Mit 8 Bausteinen erhält man damit 64 KByte.

Die Auswahl der Bausteine erfolgt wie schon bei der SBC2 mit Hilfe eines Dekoders (IC10). Der Datenbus ist durch den bidirektionalen Bustreiber 74LS245 (IC9) vom internen Bus getrennt. Der Bustreiber wird durch zwei Signale angesteuert. DIR bestimmt die Signalfluß-Richtung. Liegt DIR auf High, so werden alle Signale vom A-Teil zum B-Teil durchgeschaltet, liegt DIR auf Low, so ist es umgekehrt. Zusätzlich muß der Eingang -CS einen Low-Pegel haben, sonst sind alle Datenleitungen des Bausteins im Tri-State-Zustand, also offen.

Der DIR-Eingang ist direkt mit dem -RD-Signal verbunden. Wenn also ein Lesevorgang stattfindet und somit -RD auf Low liegt, wird der Treiber von B nach A durchgeschaltet, sofern auch -CS ein Low-Signal besitzt. Und dies ist immer dann der Fall, wenn die Karte adressiert wird. Die Selektion erfolgt mit Hilfe des 74LS85 (IC12) und der Brücken JMP2. Das IC vergleicht die Signale der A-Seite mit der B-Seite. Mit den Brücken JMP2 stellt man einen zu vergleichenden Wert ein. Bei Übereinstimmung liegt am „= OUT" ein High-Pegel an, wenn auch der „= IN" einen High-Pegel hat. Dorthin führt aber das BANKSEL-Signal, das wir bisher noch nicht verwendet haben. Ein Widerstand (R5) sorgt dafür, daß bei offenem Eingang ein High-Pegel vorliegt. Der Ausgang des Vergleiches gelangt dann an ein paar Gatter, die dafür sorgen, daß der -CS-Eingang von IC9 genau dann auf Low geht, wenn der Ausgang von IC12 auf High liegt und -MREQ auf Low liegt, also ein Speicherwunsch vorliegt.

Abb. 4.4.1 Schaltplan der Baugruppe ROA64

Die Brücken JMP1 geben einem die Möglichkeit auch 2K × 8 Speicher zu verwenden, doch das ist nicht der Normalfall. *Abb. 4.4.2* zeigt verschiedene Kombinationsmöglichkeiten für die Brückeneinstellung.

Mit JMP2 stellt man die Adresse der Speicherkarte ein, bei der sie angesprochen wird. Eine eingezeichnete Linie entspricht einer eingesteckten Brücke.

Wenn man die Z80-CPU verwendet und keine BANK/BOOT-Karte hat, so bleiben alle Brücken offen. Man erhält dann die Adresse FXXXX. Die Werte XXXX stehen für die Adreßleistungen A0 bis A15, die durch den Prozessor bestimmt werden. „F" steht für die Adreßleitungen A16 bis A19. Die Z80-Vollausbau-CPU hat keine Ausgänge für die Adressen A16 bis A19. Daher sind sie auf dem Bus offen. Offene Leitungen haben bei TTL-ICs aber den

16 / 64	JMP 1
16 / 64	für 8Kx8 RAMs
16 / 64	für 2Kx8 RAMs

JMP2	
A16 … A19	A16 … A19
0	8
1	9
2	A
3	B
4	C
5	D
6	E
7	F

Abb. 4.4.2 Brücken auf der ROA64

Abb. 4.4.3 Die Lötseite der Leiterplatte ROA64

Abb. 4.4.4 Die Bestückungsseite der Leiterplatte ROA64

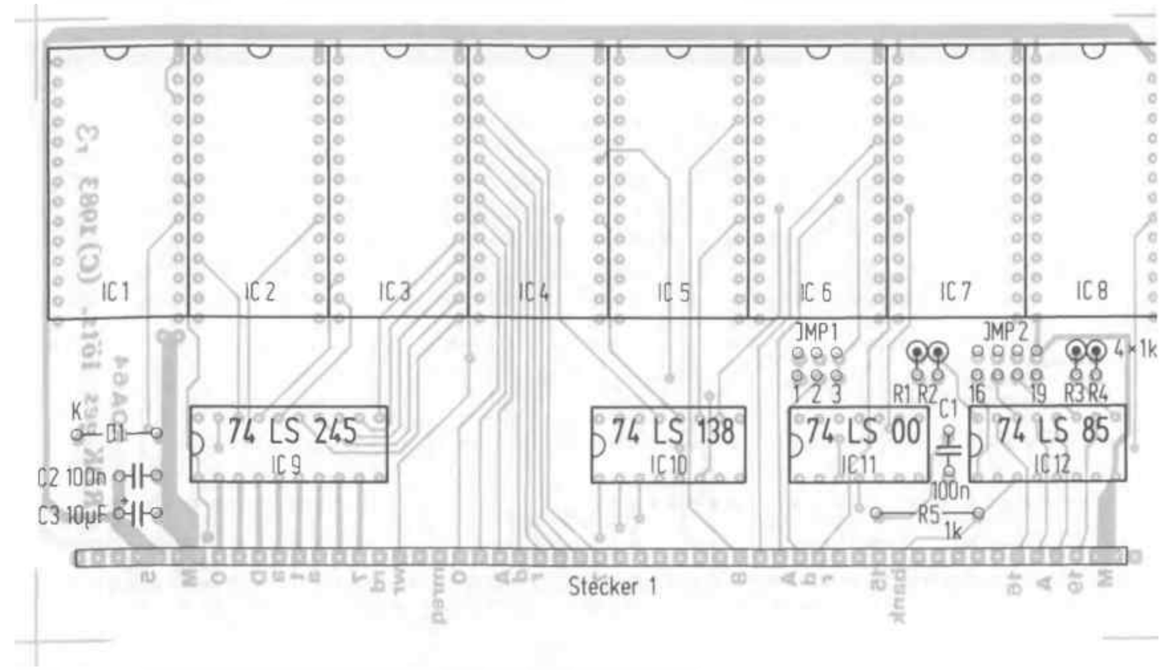


Abb. 4.4.5 Der Bestückungsplan der Baugruppe ROA64

Wert 1, daher ergibt sich binär 1111 für A16 bis A19, oder F in sedezimaler Schreibweise. Die Karte wird also von F0000 bis FFFFF angesprochen.

Arbeitet man mit der BANK/BOOT-Karte, so kann man auch andere Kombinationen verwenden. Soll die ROA64 zum Beispiel im Bereich A0000 bis AFFFF angesprochen werden, so finden Sie in *Abb. 4.4.2* unter „A“ die entsprechende Brückenbelegung für den JMP2, für den Bereich 20000 bis 2FFFF nehmen Sie die Kombination bei „2“. *Abb. 4.4.3* zeigt die Lötseite der Leiterplatte ROA64 und und *Abb. 4.4.4* die Bestückungsseite. *Abb. 4.4.5* zeigt den Bestückungsplan der Baugruppe.

Eine Liste aller benötigten Bauteile finden Sie in *Tabelle 4.4.1*.

Inbetriebnahme und Test:

Um die Baugruppe testen zu können benötigen Sie die Z80-CPU-Baugruppe oder die Baugruppe mit dem HD64180, KEY und die GDP64 (der Zusammenbau wird erst später erklärt).

Der Z80 wird mit einem Betriebsprogramm, dem Grundprogramm, ausgerüstet. *Abb. 4.4.6* zeigt die Anordnung von EPROMs und RAM auf der ROA64. Das EPROM mit dem Grundprogramm kommt ganz links in den Sockel, also in die Position IC1. Zum Betrieb wird ferner mindestens ein 8K × 8 RAM benötigt, das in die Position IC5 kommt.

Beim Betrieb ohne BANK/BOOT-Karte werden alle Brücken offengelassen. Nach dem Einschalten muß sich das Grundprogramm auf dem Bildschirm melden (Achtung, Einsteiger, lesen Sie zunächst in Kapitel 5 weiter und tun Sie so, also ob Sie eine SBC2 hätten. Die CPU-Z80 bildet zusammen mit der ROA64 im Prinzip eine SBC2 mit erweiterten Möglichkeiten, testen Sie

Tabelle 4.4.1 Die Stückliste zur Baugruppe ROA64

IC9 74LS245, bidirektionaler Datenbustreiber
IC10 74LS138, 1 aus 8 Dekoder
IC11 74LS00, Nand-Glieder
IC12 74LS85, Vergleicher
8× 28polige Fassung
1× 20polige Fassung
2× 16polige Fassung
1× 14polige Fassung
D1 Zenerdiode ZY 5.1 oder Diode 1N4002
C1, C2 100 nF
C3 10 µF, 16 V
R1, R2, R3, R4, R5 1 kΩ, 1/8 W, Wert sehr unkritisch (auch 4.7 kΩ möglich).
St1 1× 36polige und 1× 18polige Stiftleiste, gewinkelt
JMP1 doppelreihige Stiftleiste, gerade
4× Shuntstecker
1× Leiterplatte ROA64
Optional je nach Anwendung:
EPROMS von Typ 2764 mit 250 ns oder schneller. RAMs vom Typ HM 6264 P-15, oder TC 5565 P-15, oder äquivalente.

dann erst KEY und GDP64, wie dort beschrieben. Im Fehlerfall kann es aber auch an der ROA64 liegen.). Die Plätze bei IC2, IC3 und IC4 sind für weitere Programme reserviert, die im Softwarekapitel näher erklärt werden. Die drei freien Plätze bei IC6, IC7 und IC8 können zusätzlich mit RAM belegt werden.

Achtung, wenn man die HD64180-Baugruppe verwendet, muß man ggf. die Brücken JMP2 alle einsetzen, siehe Beschreibung in Kapitel 4.3.

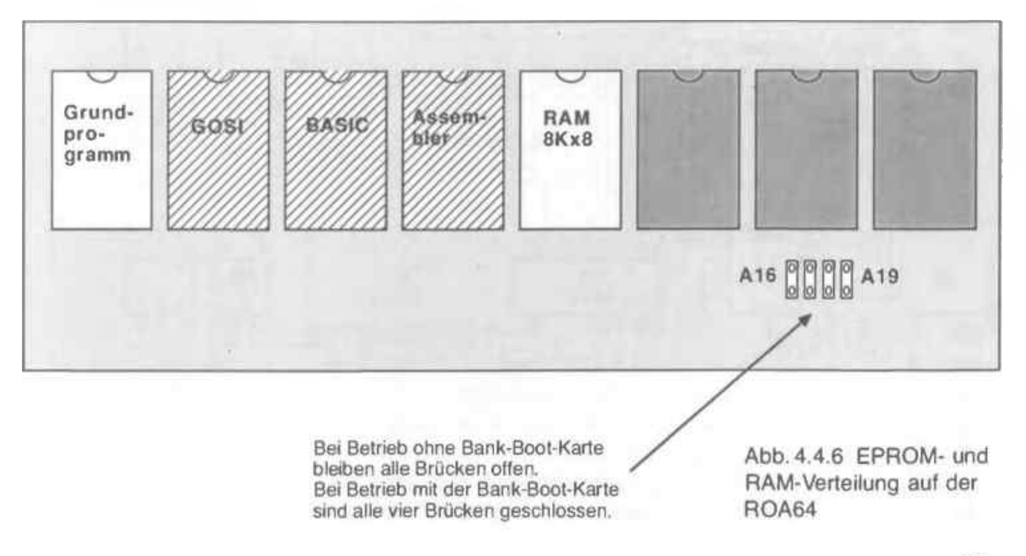


Abb. 4.4.6 EPROM- und RAM-Verteilung auf der ROA64

Kenndaten:

Spannung: + 5 V; Stromaufnahme leere Baugruppe: 120 mA; Stromaufnahme mit 4 × EPROM 2764 + 1 × RAM: 200 mA (weitere RAMs haben kaum Einfluß auf die Stromaufnahme, da es CMOS-Bausteine sind).

4.5 Die Bank-Boot-Baugruppe

Wer noch mehr Speicher haben will, der benötigt die Bank-Boot-Karte. Damit ist der Z80 in der Lage, einen Adreßraum von 1 MByte (1 Megabyte = 1024 KByte) zu bedienen.

Ferner kann sie noch eine andere Aufgabe erledigen, die für CP/M Voraussetzung ist: den „Boot". CP/M selbst benötigt nämlich einen durchgehenden RAM-Bereich von Adresse 0 bis FFFF. Ab 0 muß aber nach einem Reset ein Programm stehen, sonst weiß der Z80 nicht, was er tun

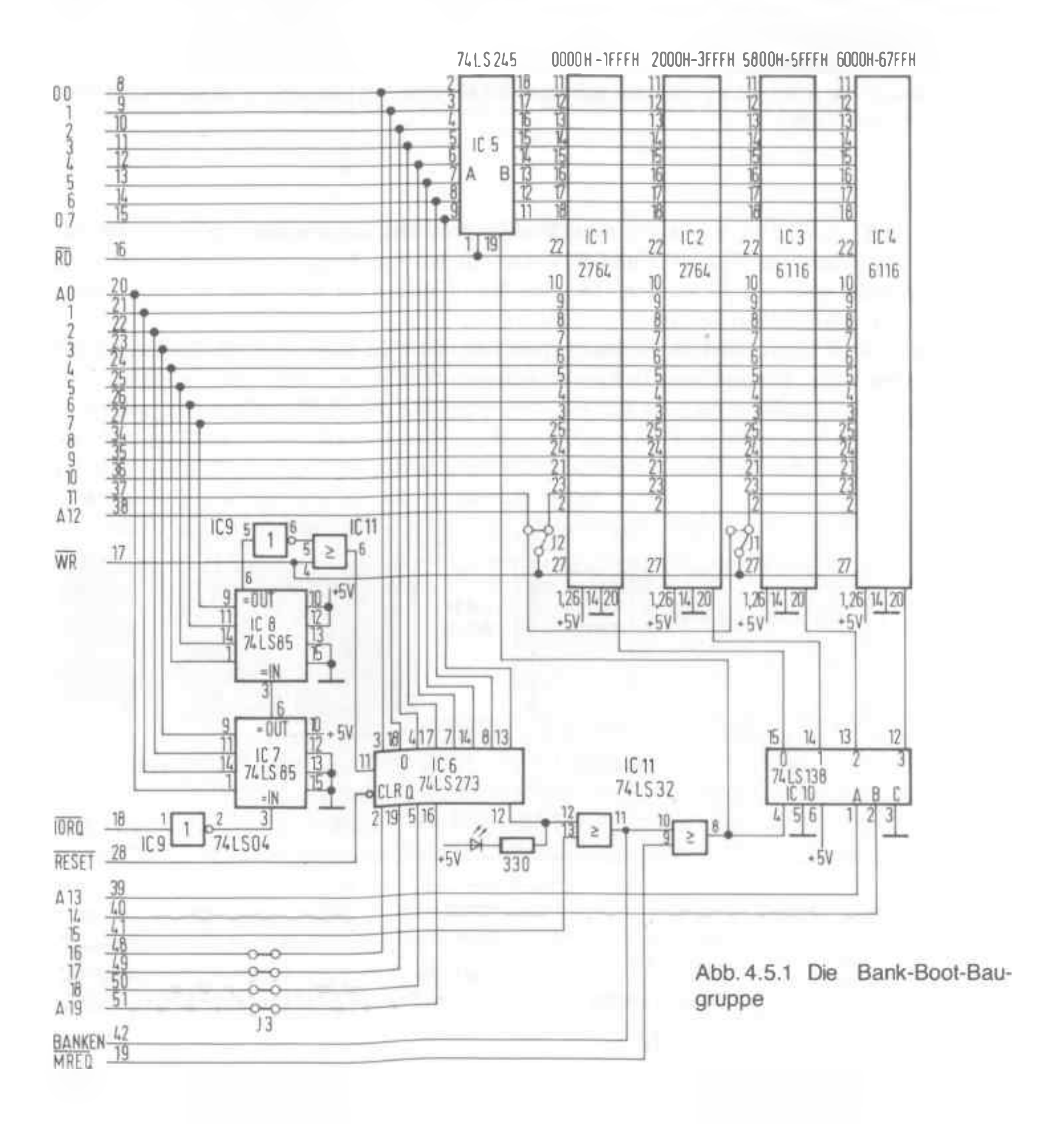


Abb. 4.5.1 Die Bank-Boot-Baugruppe

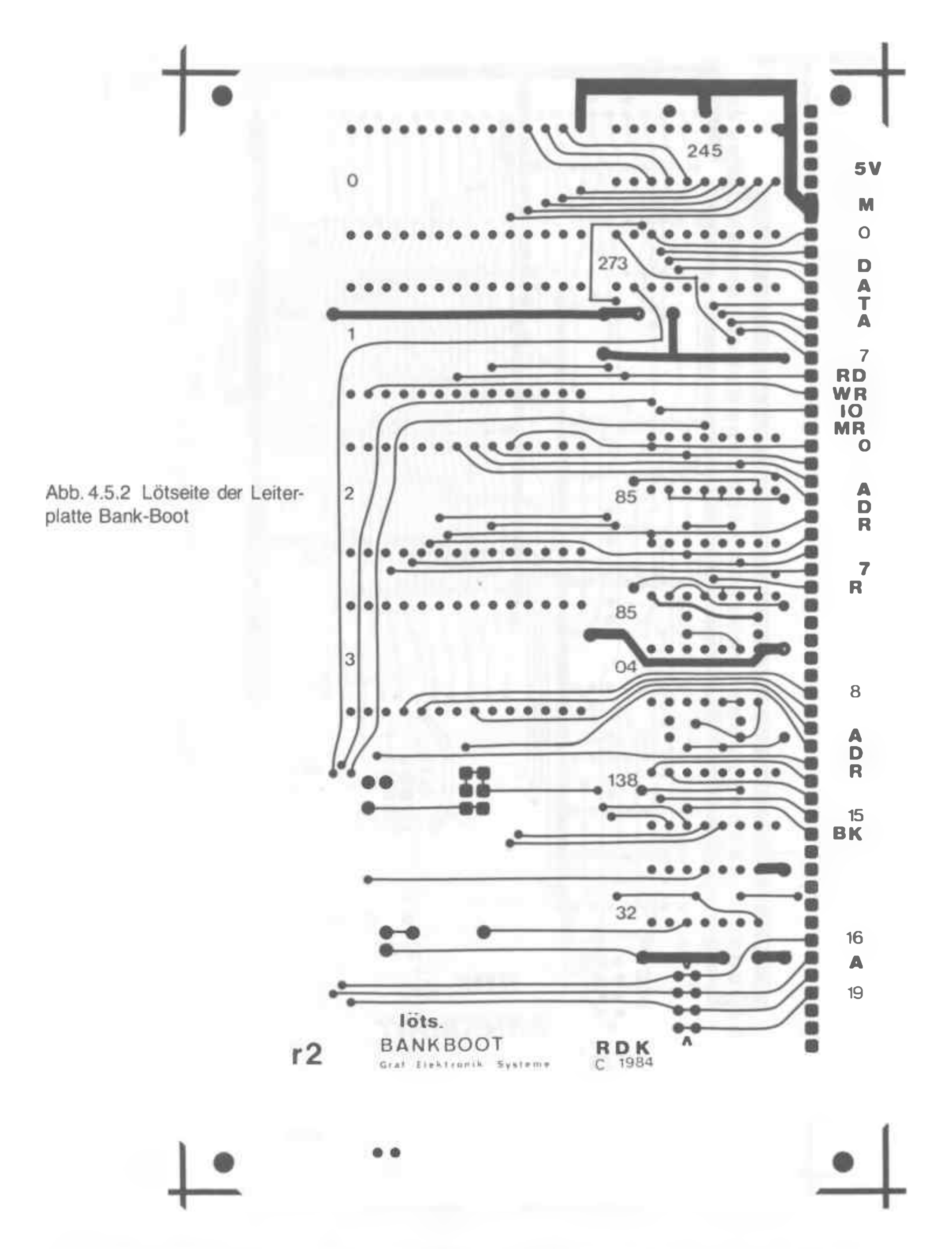


Abb. 4.5.2 Lötseite der Leiterplatte Bank-Boot

soll. Die Boot-Karte ist nun in der Lage, wahlweise in den unteren Speicherbereich RAM oder ROM einzublenden. *Abb. 4.5.1* zeigt die Schaltung, *Tabelle 4.5.1* die Stückliste und *Abb. 4.5.2* die Löstseite, *Abb. 4.5.3* die Bestückungsseite, und *Abb. 4.5.4* den Bestückungsplan.

IC5 ist ein Bustreiber. Er dient der Bustrennung und Erhöhung des Fan Outs. IC1 und IC2 sind 8-KByte-EPROMs und IC3 und IC4 sind RAMs mit je 2 KByte (bzw. 8 KByte), die als eigener

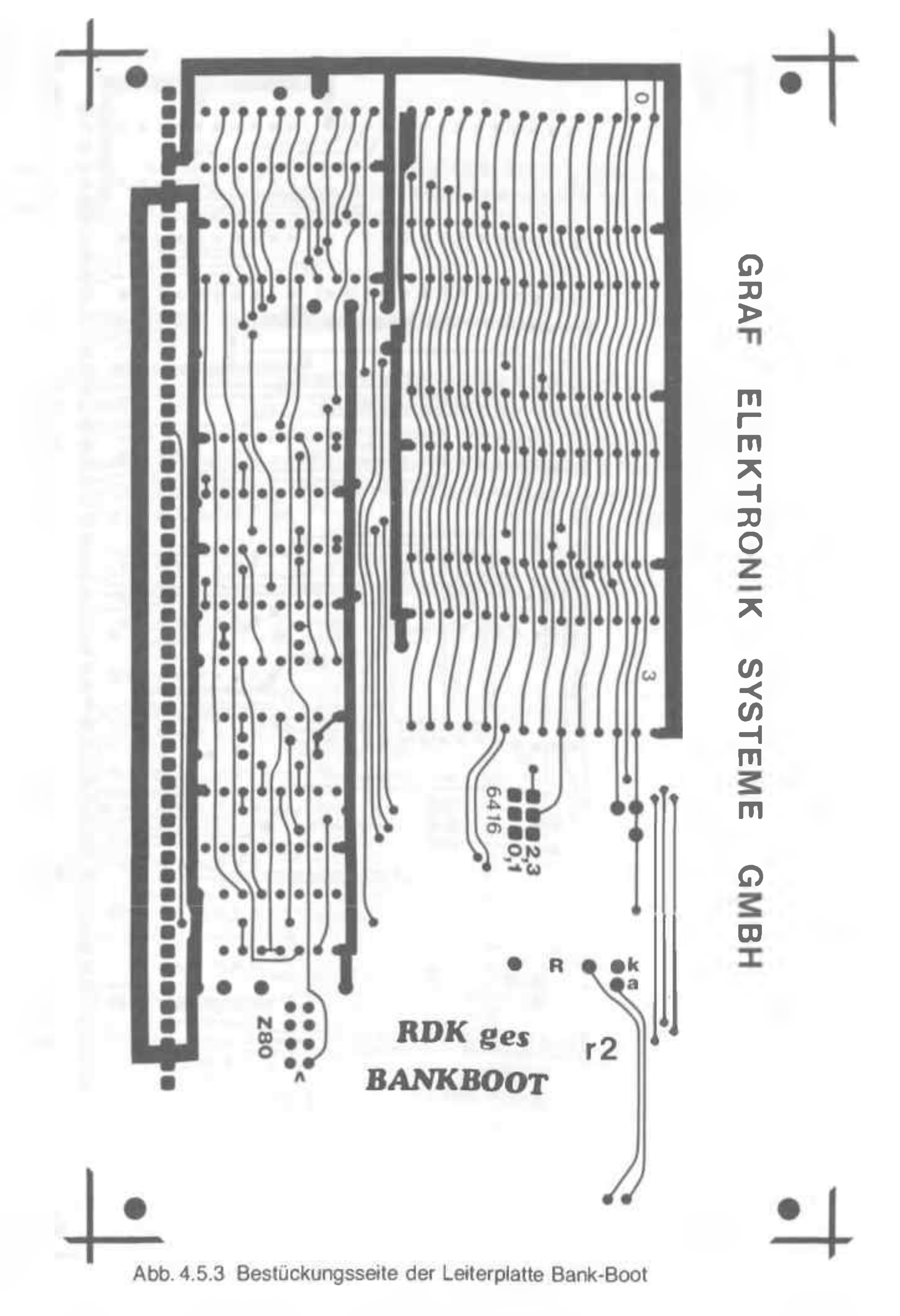


Abb. 4.5.3 Bestückungsseite der Leiterplatte Bank-Boot

freier Speicher von manchen System-EPROMs benötigt werden. Mit den Brücken J1 und J2 kann man die Bausteintypen einstellen; auf dem Layout sind jedoch die im Schaltplan angegebenen Positionen schon vorverdrahtet.

IC7 und IC8 übernehmen die Decodierung des Ports. Die Port-Adresse der Baugruppe liegt fest auf C8h. Mit IC10 wird der Speicher-Bereich von 0 bis 7FFF decodiert. Dieser Bereich ist nur

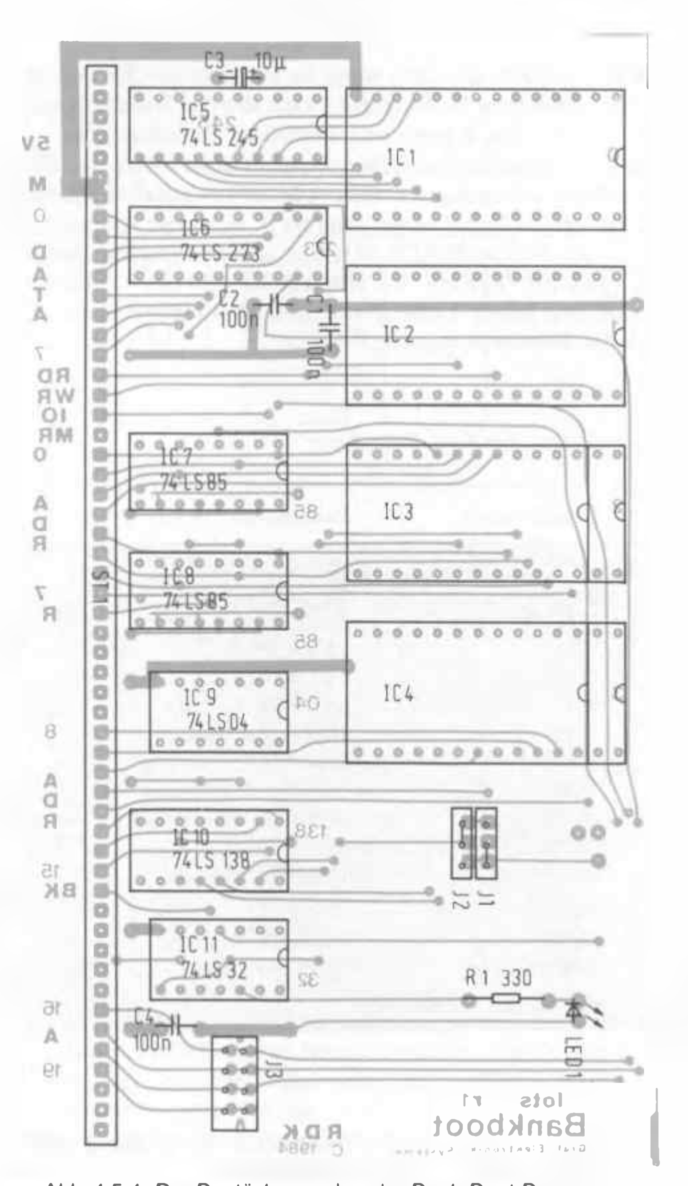


Abb. 4.5.4 Der Bestückungsplan der Bank-Boot-Baugruppe

dann aktiviert, wenn an PIN 12 des IC6 (74LS273) ein 0-Signal ausgegeben wird. Wenn dieses Bit auf 1 geht, wird die Baugruppe abgeschaltet. Mit den Bits 0 bis 3, die am Baustein IC6 liegen, wird der Bankbereich angegeben. Dazu sind dessen Ausgänge direkt mit den Adreßleitungen A16 bis A19 verbunden.

Bei der Brücke J3 kann man diese Leitungen aber auch unterbrechen, wenn man die Baugruppe zum Beispiel mit dem 68008 verwenden will, der die Adreßleitungen A16 bis A19 selbst treiben kann.

IC6 wird durch IC7 und IC8 selektiert, wenn die I/O-Adresse C8h anliegt. *Abb. 4.5.4* zeigt nochmals die genaue Bitbelegung des Ports. Bit 7 kann also den Speicherbereich der Bank-Boot-Karte ausblenden, und Bit 0 bis 3 geben eine zusätzliche Bankadresse an.
Die Leitung BANKEN, die an den Bus geführt ist, dient dazu, andere Speicherbaugruppen ein- oder auszublenden. Diese Leitung liegt normalerweise auf 1, wenn alle Speicher aktiviert werden. Die Bank-Boot-Karte legt das Signal immer dann auf 0, wenn eine Adresse kleiner 8000h, als 0 bis 7FFF anliegt UND das Bit 7 am Port C8 auf 0 liegt. In dieser Situation blendet sich die Karte in alle Bankbereiche von 0 bis 7FFF ein. Bei einem Reset wird das IC6 über den CLR-Eingang gelöscht und alle Ausgänge führen 0. Damit wird die Bank 0 angewählt, und gleichzeitig blendet sich die Baugruppe im Adreßbereich 0 bis 7FFF ein.

Tabelle 4.5.1 Die Stückliste zur Bank-Boot-Karte

Stück	Bezeichnung	Bauelement
1	IC5	74 LS 245
1	IC6	74 LS 273
2	IC7, IC8	74 LS 85
1	IC9	74 LS 04
1	IC10	74 LS 138
1	IC11	74 LS 32
4	SO28	28polige IC-Fassung
2	SO20	20polige IC-Fassung
3	SO16	16polige IC-Fassung
2	SO14	14polige IC-Fassung
3	C1, C2, C4	100 nF
1	C3	10 µF (Elko)
1	R1	330 Ω
1	LED1	Leuchtdiode
1	ST1 (Stecker)	18- und 36polige Steckerleiste
1	Platine mit Lötstoplack	

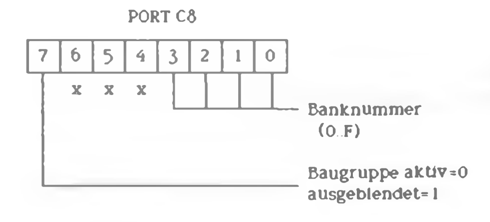
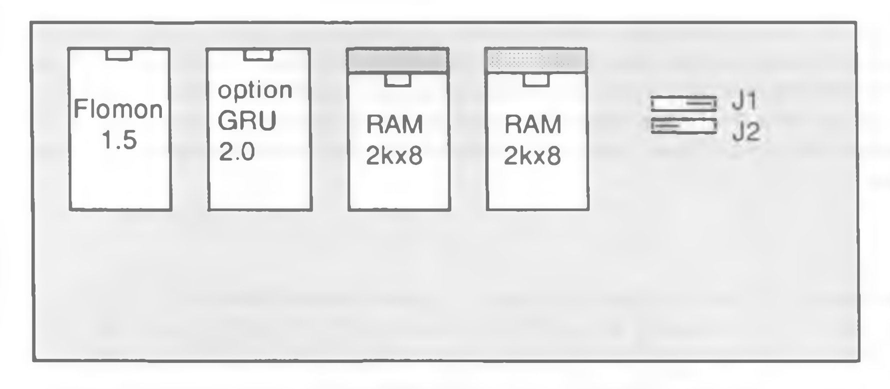


Abb. 4.5.5 Die Belegung des Ports C8. Durch Einschreiben von Werten in diesen Port wird vom Z80 aus 1 MByte Speicher verwaltet



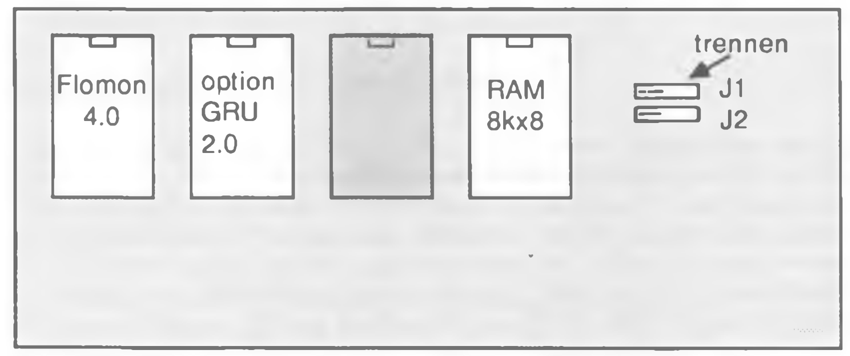


Abb. 4.5.6 Die Bestückung mit EPROMs und RAMs, für FLOMON 1.5 und für FLOMON 4.0 (neue Version siehe Kapitel 8)

Boot

Ein Programm auf dieser Baugruppe, das Boot-Programm, sorgt nun dafür, daß ein Teil des EPROM-Bereichs in den allgemein zugänglichen RAM-Bereich ab F000 kopiert wird und dort angesprungen wird. Dieses Programm blendet den Speicher der Bank-Boot-Karte wieder aus und sorgt seinerseits für die weiteren Funktionen des Computers, zum Beispiel unter CP/M. *Abb. 4.5.6* zeigt die Anordnung von EPROMs und RAMs auf der Bank-Boot-Baugruppe. Das neue Programm zum Betrieb von CP/M nennt sich Flomon 4.0 und sitzt ganz links. Es benötigt einen RAM-Baustein mit 8 KByte auf dem rechten Steckplatz. Wahlweise kann man auf dem verbleibenden zweiten Steckplatz von links ein Grundprogramm stecken, das dazu auf Adresse 2000h übersetzt ist. Beide EPROMs sind im Handel erhältlich. Das Grundprogramm kennen Sie vielleicht von der SBC2 her. Man kann damit viele Tests ausführen, wenn es Schwierigkeiten geben sollte (Grundprogramm siehe Kap. 5).

Abb. 4.5.7 zeigt die gesamte Anordnung. Neben der Bank-Boot-Baugruppe und der Vollausbau-CPU benötigt man noch eine Speicherbaugruppe mit 64 KByte, und natürlich die FLO2-Baugruppe, die die Floppy-Ansteuerung übernimmt. Weitere Speicherkarten können hinzugefügt werden, zum Beispiel um CP/M3.0 zu fahren oder eine RAM-Floppy zu betreiben. Die erste RAM-Baugruppe wird auf die Bank 0 eingestellt, also alle Brücken A16 bis A19 werden eingesetzt.

Abb. 4.5.8 zeigt mögliche Erweiterungen. Auf der Bank E0000 können zusätzlich das Grundprogramm und/oder andere Sprachen untergebracht werden. Von 0 bis DFFFF kann man RAM unterbringen, oder von 0 bis FFFFF, wenn man auf EPROM-Plätze verzichtet.

Auf der Bank-Boot-Karte sitzen Flomon und der RAM-Speicher. Im Flomon gibt es Unterprogramme, mit welchen man Daten in 128-Byte-Blöcken von einer Bank zur anderen verschieben kann.

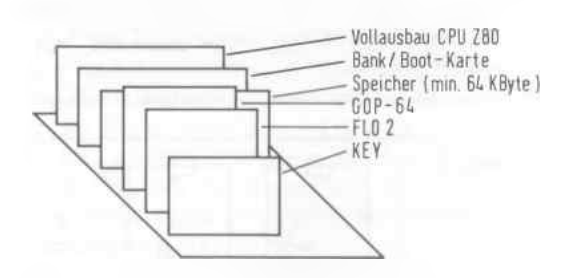


Abb. 4.5.7 CP/M-80-Konfiguration

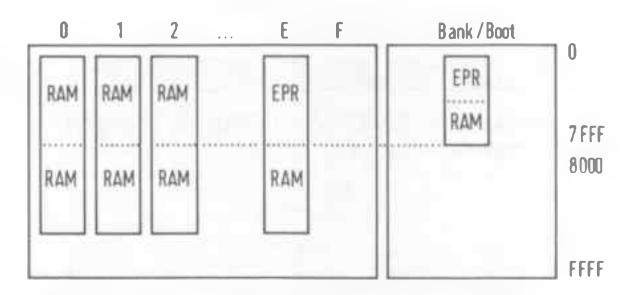


Abb. 4.5.8 Alle Bänke in einer Übersicht

5 Bildschirm und Tastatur

In diesem Kapitel wollen wir den Computer als Ganzes in Betrieb nehmen. Wir werden ihn zunächst in den ersten beiden Abschnitten mit dem Nötigsten ausstatten. Wir lernen das Grundprogramm kennen und werden einige erste Kontakte mit Computerprogrammen haben.

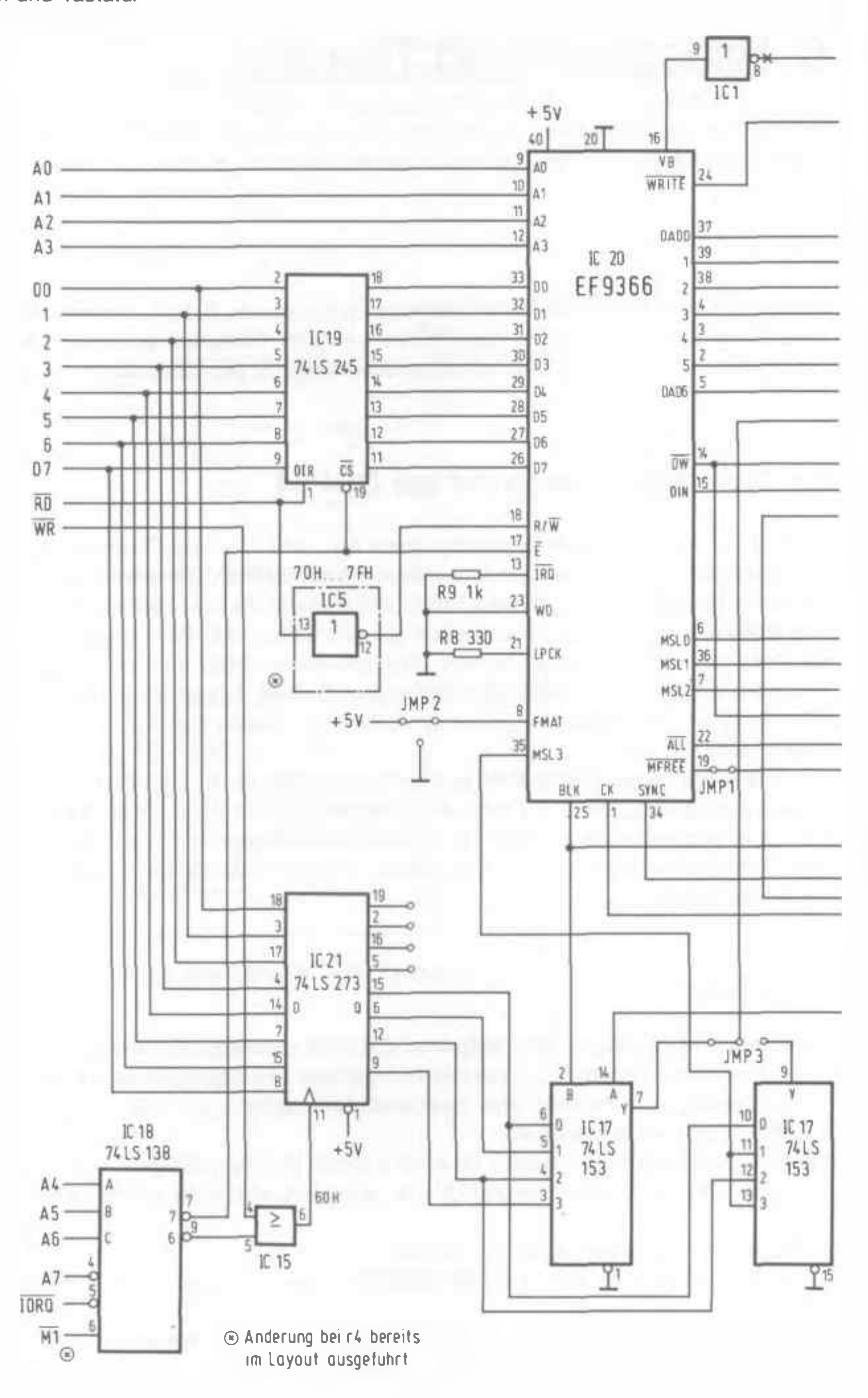
5.1 Schreiben lernen mit der GDP64

Vielleicht ist mancher schon ungeduldig geworden, weil er seinen Computer noch nicht richtig programmieren konnte. Zunächst aber sollten Sie ein Gefühl dafür bekommen, daß die Mikroelektronik Intelligenz in die Technik bringt und überall schaltend und regelnd eingreifen kann. Jetzt werden die technischen Voraussetzungen geschaffen, daß Ihr Computer sich „schriftlich" mit Ihnen unterhalten und Ihre Befehle entgegennehmen kann. In diesem Kapitel werden ein Tastatur-Interface und ein Bildschirm-Endgerät aufgebaut. Dieses letzte Gerät kann Ihnen vom Blümchen bis zum Schaltplan alles auf den Bildschirm malen, was das Herz begehrt. Schreiben kann es natürlich auch.

Auf der Leiterplatte GDP64K befinden sich alle Bauteile, die zum Betrieb einer Bildschirmsteuerung nötig sind. *Abb. 5.1.1* zeigt die umfangreiche Schaltung, die nach und nach aufgebaut wird. Auf eine vollständige Darstellung muß hier allerdings verzichtet werden. *Tabelle 5.1.1* zeigt die Stückliste und *Abb. 5.1.2* die Lötseite, *Abb. 5.1.3* die Bestückungsseite und *Abb. 5.1.4* den Bestückungsplan.

So wird aufgebaut

1. Einlöten aller Fassungen. Bitte aufpassen und nicht versehentlich 14polige Fassungen anstelle von 16poligen einlöten. Man kann falsch eingelötete Fassungen praktisch nicht mehr auslöten. Dazu benötigt man entweder eine sogenannte Lötsauglitze oder eine Entlötpumpe. Also bitte vorher lieber zweimal schauen.
2. Einlöten aller diskreten Bauteile. Diskrete Bauteile sind zum Beispiel: Widerstände, Kondensatoren, Dioden, Transistoren und Quarze, also alles, was nicht mehrere Elemente integriert hat.
3. Einlöten der 36poligen Stiftleiste (Stecker 1).
4. Alle ICs einsetzen bis auf die RAM-Bausteine (4164 o. ä.) und das IC EF9366, den Grafik-Prozessor.
5. Nun kann man die Karte auf den Bus stecken. Die POW5V wird ebenfalls auf den Bus gesteckt. Die SBC2-Baugruppe wird noch nicht eingesteckt!
6. Einschalten und Messen. An Pin 8 des IC5 (7404 beim Quarz) muß ein 14-MHz-Takt-Signal anliegen. Am Prüfstift leuchten alle vier LEDs (W1, W2, H und L). Mit einem Oszilloskop kann man diese Frequenz messen (wenn es gut genug ist).


IC1
IC 20
EF9366
IC19
74 LS 245
IC5
70H...7FH
R9 1k
R8 330
JMP 2
JMP1
JMP 3
IC 21
74 LS 273
IC 17
74 LS
153
IC 18
74 LS 138
IC 15
Änderung bei r4 bereits
im Layout ausgeführt

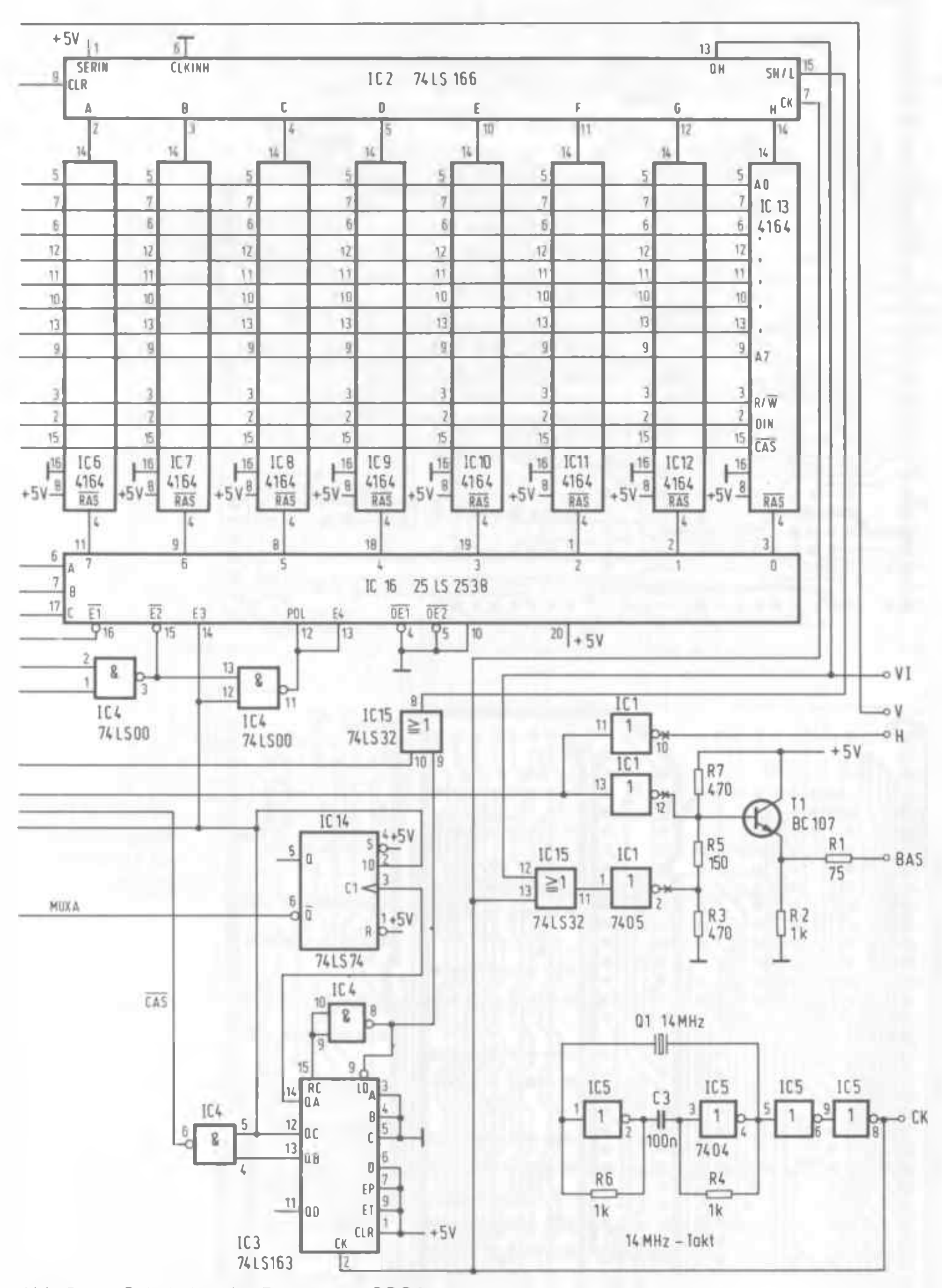


Abb. 5.1.1 Schaltplan der Baugruppe GDP64

Abb. 5.1.2 Die Lötseite der Leiterplatte GDP64

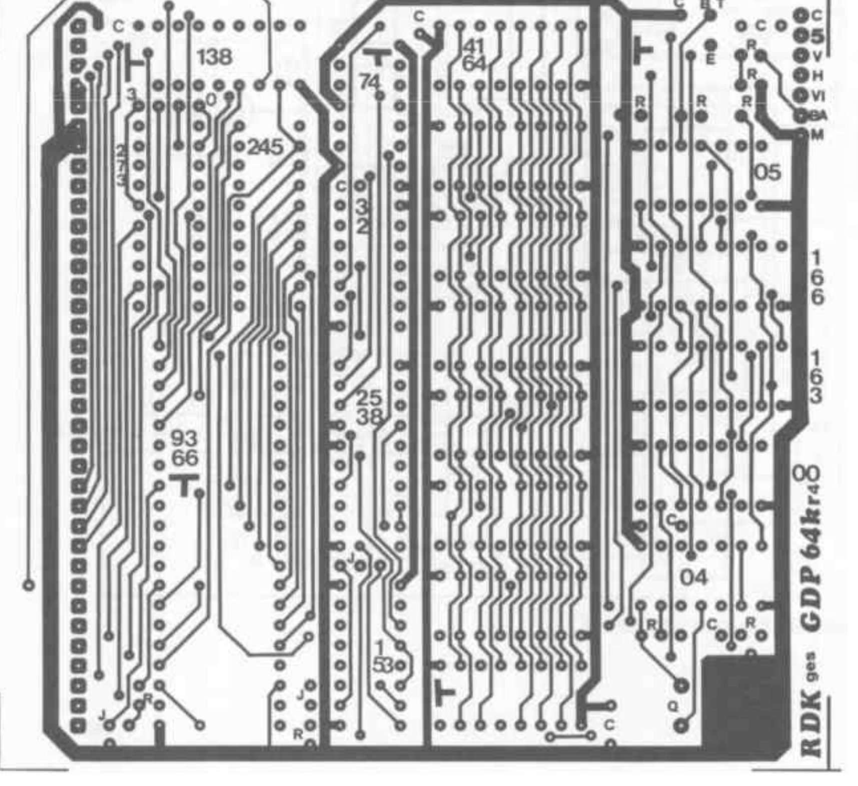


Abb. 5.1.3 Die Bestückungsseite der Leiterplatte GDP64

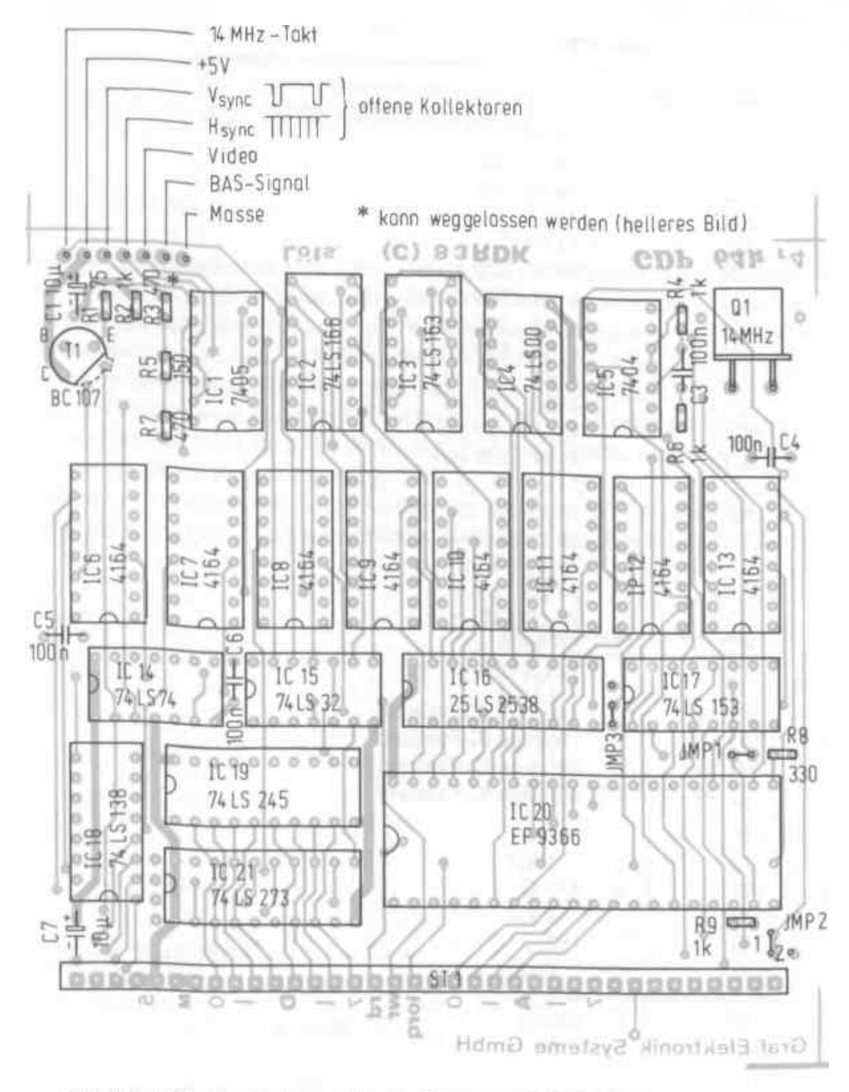


Abb. 5.1.4 Der Bestückungsplan der Baugruppe GDP64

7. Messen an Pin 1 der Fassung des EF9366 (IC1). Dort muß auch ein Takt von 1,75 MHz anliegen. Beim Prüfstift sieht man alle LEDs leuchten.
8. Spannung abschalten und das IC EF9366 einsetzen. Dabei auf die Orientierung der Nase achten, sie muß in Richtung der ICs 19 und 21 zeigen. (Bestückungsplan!) Vorsicht, das IC ist teuer!
9. Spannung einschalten und an Pin 34 des EF9366 messen. Beim Prüfstift leuchten W1 und W2, die LED H ist dunkler und die LED L hell.

Tabelle 5.1.1 Die Stückliste zu GDP64

Stück	Benennung	
1	JC1	74 05
1	JC2	74 LS 166
1	JC3	74 LS 163
1	JC4	74 LS 00
1	JC5	74 04
1	JC6	4164 · 200 ns
1	JC7	4164 · 200 ns
1	JC8	4164 · 200 ns
1	JC9	4164 · 200 ns
1	JC10	4164 · 200 ns
1	JC11	4164 · 200 ns
1	JC12	4164 · 200 ns
1	JC13	4164 · 200 ns
1	JC14	74 LS 74
1	JC15	74 LS 32
1	JC16	25 LS 2538
1	JC17	74 LS 153
1	JC18	74 LS 138
1	JC19	74 LS 245
1	JC20	EF 9366
1	JC21	74 LS 273
5	SO 14	14polige IC-Fassung
12	SO 16	16polige IC-Fassung
3	SO 20	20polige IC-Fassung
1	SO 40	40polige IC-Fassung
1	R1	75 Ω
1	R2	1 kΩ
1	R3	470 Ω
1	R4	1 kΩ
1	R5	150 Ω
1	R6	1 kΩ
1	R7	470 Ω
1	R8	330 Ω
1	R9	1 kΩ
1	C1	10 µF
1	C2	100 nF
1	C3	100 nF
1	C4	100 nF
1	C5	100 nF
1	C6	100 nF
1	C7	10 µF
1	T1	BC 107
1	Q1	14,00 MHz
1	Stecker 1 36polig	
1	Platine mit Lötstopplack	

Mit dem Oszilloskop kann man sich die Pulsform genauer ansehen. Es handelt sich um das HSYNC-Signal, das später erklärt wird.

10. Messen an Pin 16. Mit dem Prüfstift erkennt man, daß W1 und W2 sichtbar flimmern. Die LEDs H und L leuchten auch. Es handelt sich hierbei um das VSYNC-Signal.

Daten auf den Bildschirm

Nun kann man einen Bildschirm anschließen. Dabei gibt es sehr verschiedene Möglichkeiten. Am einfachsten ist der Anschluß an einen Video-Monitor. Dieser besitzt einen BAS-Eingang, den man direkt mit dem Anschluß BAS der GDP64-Baugruppe verbinden kann. Dazu besitzt die Baugruppe am Platinenrand eine Lochreihe, deren Belegung *Abb. 5.1.5* zeigt. BAS heißt Bild-, Austast- und Synchronsignal. *Abb. 5.1.6* zeigt das Verbindungsschema.

Dann gibt es Fernsehgeräte mit einem AV-Eingang. *Abb. 5.1.7* zeigt die Belegung eines AV-Steckers. Auch dorthin kann man das BAS-Signal direkt führen, meist ist jedoch noch ein Widerstand von 75 Ω zur Anpassung nötig.

Und dann gibt es natürlich noch die TV-Geräte, die nur einen HF-Eingang (Antenneneingang) besitzen. Und um so ein Fernsehgerät anschließen zu können, benötigt man einen HF-Modulator.

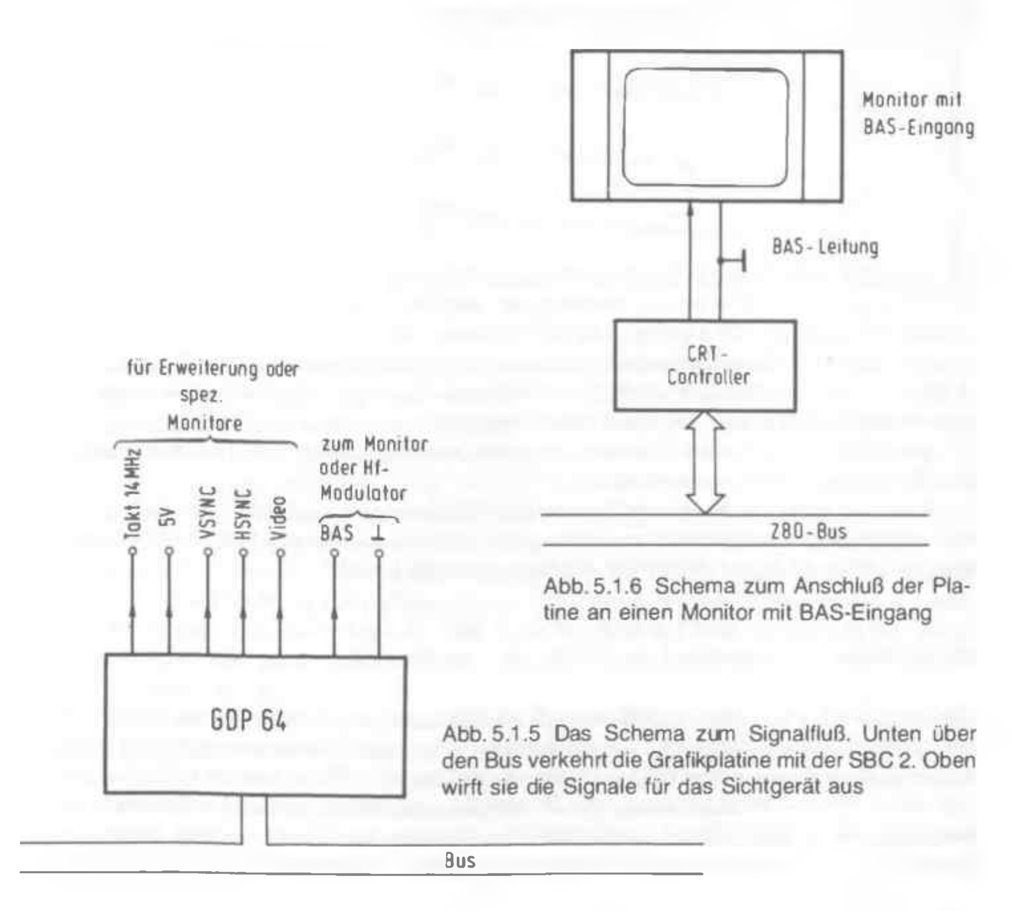


Abb. 5.1.6 Schema zum Anschluß der Platine an einen Monitor mit BAS-Eingang

Abb. 5.1.5 Das Schema zum Signalfluß. Unten über den Bus verkehrt die Grafikplatine mit der SBC 2. Oben wirft sie die Signale für das Sichtgerät aus

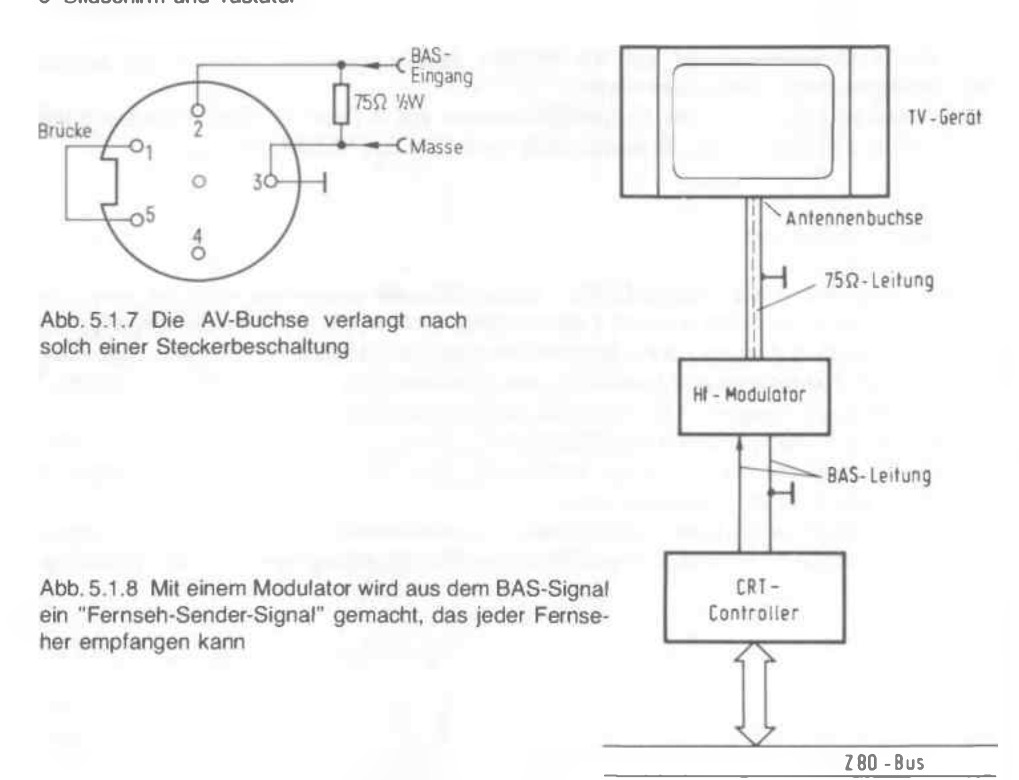


Abb. 5.1.7 Die AV-Buchse verlangt nach solch einer Steckerbeschaltung

Abb. 5.1.8 Mit einem Modulator wird aus dem BAS-Signal ein "Fernseh-Sender-Signal" gemacht, das jeder Fernseher empfangen kann

Der Anschluß sieht dann wie in *Abb. 6* aus. Der BAS-Ausgang der GDP64K-Baugruppe wird an den Eingang des HF-Modulators angeschlossen und der Ausgang des HF-Modulators an den Antenneneingang des TV-Gerätes. Der HF-Modulator benötigt noch eine 5-V-Spannung zum Betrieb, die an der Buchsenreihe der GDP64K-Karte herausgeführt ist.

Monitore liefern ein schärferes Bild als TV-Geräte. Der Preis von Monitoren bewegt sich zur Zeit um die 300 DM herum. Der Kauf lohnt sich, wenn man intensiver in die Mikrocomputertechnik einsteigen will. Vor allem kann man dann am Computer arbeiten, ohne den Rest der Familie vom Fernsehprogramm abzuschneiden.

Wenn man nach Anschluß eines Bildschirmgerätes wieder die Spannung einschaltet, so muß ein schwach sichtbarer Rahmen auf dem Bildschirm erscheinen, abhängig von der Helligkeitseinstellung am Gerät. Dann arbeitet die GDP-Karte soweit korrekt.

Wie ein Fernsehbild entsteht

Um ein Bild auf einem Fernsehbildschirm abzubilden, muß es zunächst in Zeilen zerlegt werden. Bei unseren Geräten sind das 625 Zeilen (in den USA verwendet man eine andere Zeilenzahl). Blitzschnell wird das gesamte Bild aus Zeilen von hellen und dunklen Punkten zusammengesetzt. Auf dem TV-Gerät werden dabei zuerst alle ungeraden Zeilen eingeschrieben und nach Ablauf von 20 ms alle geraden. *Abb. 5.1.9* und *Abb. 5.1.10* zeigen den Ablauf. So kann man Flimmern vermeiden. Das Verfahren wird Zeilensprungverfahren genannt, da nur jede zweite Zeile

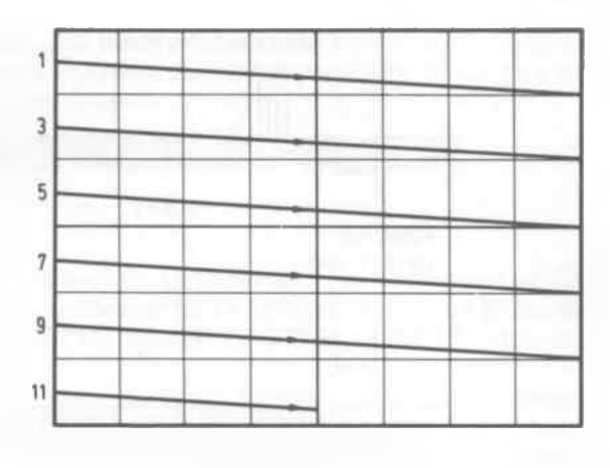


Abb. 5.1.9 Beim Zeilensprung-Verfahren wird das Bild so in zwei Teile zerlegt, daß erst die Zeilen mit ungerader Zeilennummer vom Elektronenstrahl geschrieben werden, dann die mit gerader. Dadurch erreicht man für das Auge eine hohe Auflösung bei vergleichsweise niedriger Datenübertragungsrate

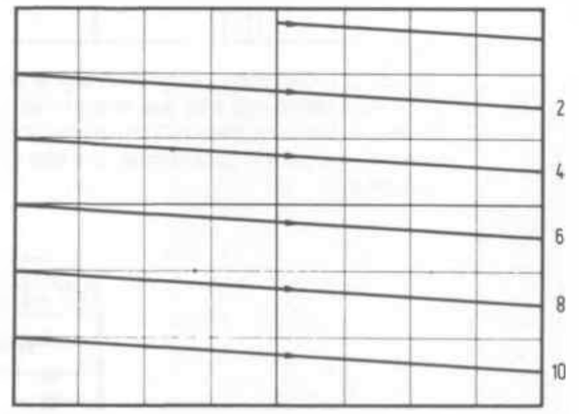


Abb. 5.1.10 Nach den ungeraden Zeilen werden die mit gerader Nummer geschrieben

geschrieben wird. Meistens wird aber bei der Erzeugung des Videosignals durch Computerelektronik das Zeilensprungverfahren nicht angewendet, sondern zweimal dasselbe „Halbbild" ausgegeben. Dadurch verringert sich der Flimmereffekt nochmals, aber man hat gegenüber einem normalen Bild nur die halbe Zeilenzahl zur Verfügung. Die GDP64-Schaltung arbeitet so.

Bei der Ausgabe zum Bildschirm genügt es nicht, die in Zeilen zerlegte Information, also das Bildsignal oder auch Videosignal genannt, einfach an den Bildschirm zu übertragen, denn der Bildschirm „weiß" ja gar nicht, wo das Bild anfängt. Dazu werden weitere Signale benötigt.

Zum einen das sogenannte Vertikal-Synchronsignal (VSYNC). Es erscheint alle 20 ms und bestimmt, wann ein Halbbild neu anfängt. Ein zweites Signal, das Horizontal-Synchronsignal, gibt an, wann eine neue Zeile beginnt. Man kann das Bild auf dem Bildschirm mit diesen Synchronsignalen sehr genau rekonstruieren. Das Horizontal-Synchronsignal, kurz VSYNC genannt, erscheint alle 64 µs.

Abb. 5.1.11 zeigt eine Zusammenfassung aller Signale. Das HSYNC- und VSYNC- sowie Video-Signal werden dann noch zu einem gemeinsamen Signal gemischt, dem BAS-Signal. Die Synchronsignale kann man im BAS-Signal von den anderen Signalen durch den Spannungswert unterscheiden. Dies geschieht im Monitor oder TV-Gerät automatisch.

Für Geräte, die getrennte Eingänge besitzen, sind die Signale aber auch getrennt auf der GDP64K-Baugruppe herausgeführt.

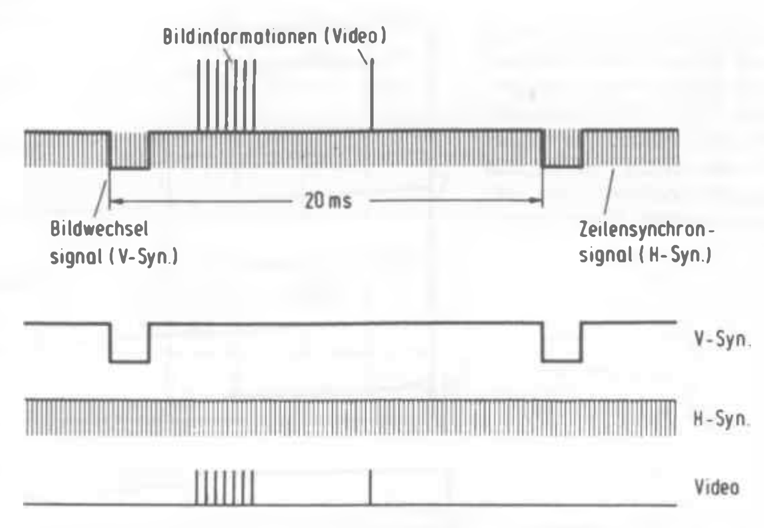


Abb. 5.1.11 Das vollständige BAS-Signal ist aus der Bildinformation (Video) ganz unten, die hier aus einigen hellen Punkten in verschiedenen Zeilen besteht, und dem H-Sync-Signal, das einen jeden Zeilenwechsel bewirkt, und dem V-Sync-Signal, das den Bildwechsel begleitet, zusammengesetzt

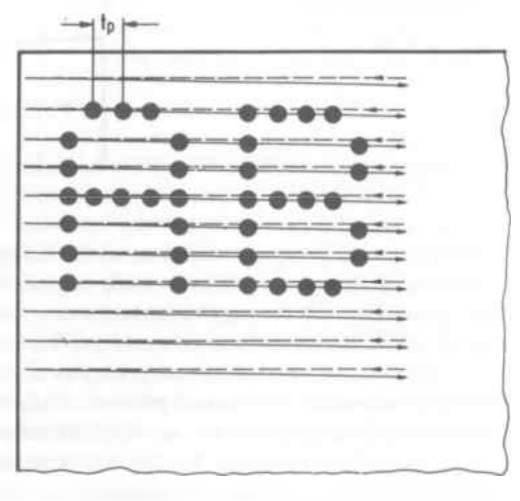


Abb. 5.1.12 Ein Buchstabe ist aus Punkten zusammengesetzt, die sich in mehreren verschiedenen Zeilen befinden, je nach Gestalt des Zeichens

Punkte werden zu Zeichen

Ein Buchstabe, der auf dem Bildschirm erscheinen soll, muß in Rasterpunkte zerlegt werden. Diese Rasterpunkte müssen dann hintereinander so ausgegeben werden, daß die Zeichen richtig geschrieben werden *(Abb. 5.1.12)*. Der Abstand der Rasterpunkte ist der sogenannte Bildpunkttakt, bei uns beträgt er 14 MHz. Dieser Takt bestimmt die Auflösung, die Schärfe des Bildes, in horizontaler Richtung. Ein Bildpunkt entspricht später einem Bit einer RAM-Zelle. Ist das Bit auf 0, so leuchtet der Punkt, ist das Bit auf 1, so bleibt der Bildpunkt dunkel. Die RAM-Bausteine

müssen dazu in der richtigen Weise adressiert werden. Genau diesen Job und das Erzeugen der Synchronsignale zur rechten Zeit und das regelrechte Einspeichern von Grafik-Bildpunkten und vieles mehr, das leistet alles der Baustein EF9366.

Experimente

1. Man verbinde PIN 14 des Sockels von IC6 mit Pin 16. Dann entsteht ein Linienmuster auf dem Bildschirm *(Abb. 5.1.13)*. Wenn man andere Pins mit Pin 14 verbindet, ergeben sich Bilder, wie in den *Abb. 5.1.14a..g* gezeigt. Durch Kombination dieser Signale läßt sich auch eine individuelle Zeile auswählen.

Die Informationen, welche Spalte ausgewählt wird, befinden sich auf den Leitungen DAD0 . . . DAD6 des Grafikprozessors EF9366, die gerade verwendet wurden. Jedoch kann man diese Information so nicht sichtbar machen. Denn die Information wird gemultiplext. Das heißt,

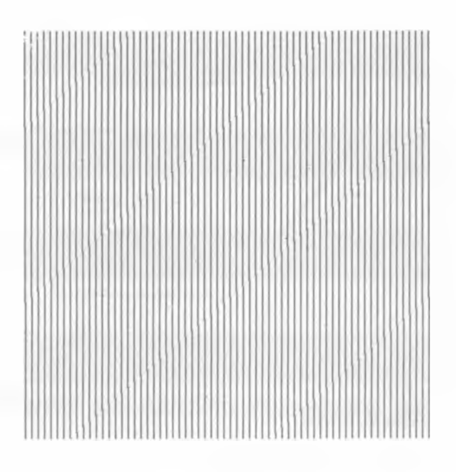


Abb. 5.1.13 Das erste Testbild, das sich durch Verbinden von Pin 14 mit Pin 16 am Sockel der Speicher-ICs ergibt

Abb. 5.1.14 a

Abb. 5.1.14 b

Abb. 5.1.14 c

Abb. 5.1.14 d

Abb. 5.1.14 e

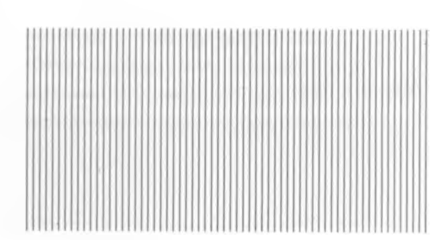


Abb. 5.1.14 f

Abb. 5.1.14 a–g Das sind die Testbilder, die entstehen, wenn man bestimmte Pins am Sockel der Speicher-ICs kurzschließt: a = 14 mit 13, b = 14 mit 10, c = 14 mit 11, d = 14 mit 12, e = 14 mit 6, f = 14 mit 7 und g = 14 mit 5

Abb. 5.1.14 g

in zeitlich kurz aufeinanderfolgenden Abständen werden zwei Informationen auf denselben Leitungen übertragen. Das ist notwendig, weil die hier verwendeten RAM-Bausteine nur sehr wenige Anschlüsse haben. Das RAM besitzt zwei Steuerleitungen: $\overline{CAS}$ und $\overline{RAS}$. Die Abkürzungen bedeuten Column-Adress-Strobe (CAS), also Signal für die Spalte, und Row-Adress-Strobe (RAS), also Signal für die Reihe. *Abb. 5.1.15* zeigt den zeitlichen Ablauf der Signale. Bei

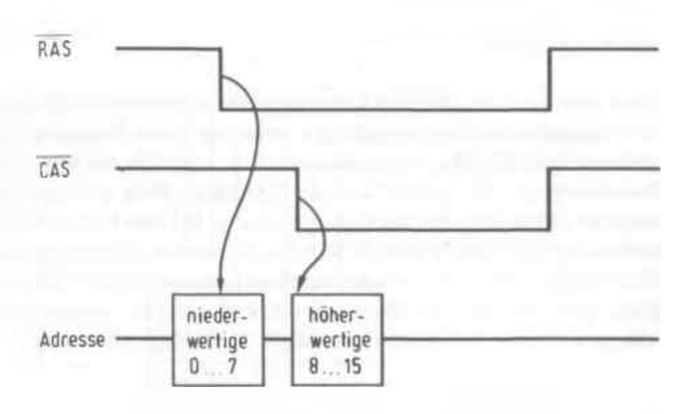


Abb. 5.1.15 Die Zeitpunkte, bei welchen die Einzelteile einer Adresse ausgegeben werden, und die Signale, die eine Übernahme durch das Speicher-IC bewirken

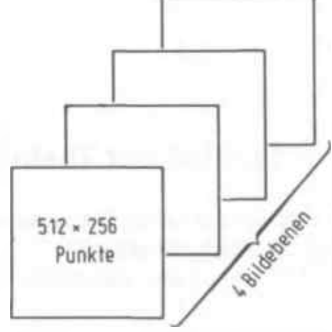



Abb. 5.1.16 Das sind die vier Seiten, die voneinander unabhängige Bilder enthalten können

fallender Flanke an Pin 4 der Speicher-ICs werden die ersten acht Adreßbits übernommen. Bei fallender Flanke an Pin 15 weitere acht.

Multiplex!

An die RAM-Speicher werden also die Adressen 0 . . . 15 übertragen, das sind 16 Leitungen. Somit kann man einen Speicher von 2^{16} = 65536 Speicherzellen adressieren. In der Baugruppe finden acht RAM-Speicher Platz, es gibt insgesamt 8 × 65536 Bits im Speicher.

Auf den Bildschirm werden 512 Punkte pro Zeile und 256 Zeilen dargestellt. Mit ihrem großen Speicherplatz kann die GDP64-Baugruppe vier solcher Bildebenen speichern, die man wahlweise auf dem Bildschirm sichtbar machen kann. Die Auswahl geschieht über die Leitung DAD7 (Pin 13 des Speicher-ICs), die nicht vom EF9366 herkommt, sondern von einem IC, das die Umschaltung der Seiten bestimmt. Die Aufteilung zeigt *Abb. 5.1.16*.

Jeder der RAM-Bausteine ist für eine Gruppe von Linien zuständig. Jetzt können diese Bausteine eingesteckt werden. Auf die Polung achten, denn sonst werden die Bausteine zerstört. Achtung für Selbstbauer, die keine Leiterplatte verwenden! Bei den RAM-Bausteinen liegt der Masseanschluß an Pin 16 und nicht an Pin 8, wie sonst üblich. Die Spannungsversorgung liegt dagegen an Pin 8.

SBC2 und GDP64

Es ist nicht möglich, Ihnen in Kürze zu erklären, wie die Platine GDP64 im einzelnen funktioniert. Es ist zunächst auch nicht wichtig – wenn Sie genau löten und nichts verkehrt einstecken, dann wird sie funktionieren. Sie werden es sehen, wenn Sie die SBC2-Baugruppe mit den zwei RAM-Bausteinen im 24poligen Gehäuse bestücken (falls sie noch nicht bei früheren Versuchen eingesteckt wurden) und das Grundprogramm in Form von zwei EPROMS 2732 einsetzen. Dabei kommt das mit 0 beschriftete EPROM in die Fassung 0 (IC6) und das mit 1 beschriftete EPROM in die Fassung 1 (IC7). Bitte vorher auf Pin 1 achten! Die SBC2-Baugruppe wird danach in die BUS-Karte gesteckt und die Spannung eingeschaltet. Es meldet sich dann das Grundprogramm. *Abb. 5.1.17* zeigt den Bildschirminhalt. Herzlichen Glückwünsch, wenn es läuft.

RDK-Grundprogramm

1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter

Abb. 5.1.17 Diese Meldung gibt SBC 2 nach dem Einschalten ab, wenn das Grundprogramm eingesteckt ist

5.2. Anschluß der Tastatur

Damit eine Tastatur an den Computer angeschlossen werden kann, wird eine weitere Baugruppe benötigt: Die KEY-Baugruppe.

Abb. 5.2.1 zeigt den Schaltplan, *Tabelle 5.2.1 die Stückliste und Abb. 5.2.2* die Lötseite,

Tabelle 5.2.1 Die Stückliste für KEY

Anzahl	Typ	Nr. im Schaltplan	
2	74 LS 86	JC1, JC2	
1	74 LS 04	JC3	
1	74 LS 74	JC4	
1	74 LS 00	JC5	
1	74 LS 245	JC6	
1	74 LS 374	JC7	
1	74 LS 32	JC8	
2	74 LS 85	JC9, JC10	
3	Kondensator	100 nF	C1, C2, C4
1	Elko	10 µF	C3
1	Steckerleiste 15polig	Stecker 1	
1	Steckerleiste 36polig	Stecker 2	
1	Netzwerkwiderstand 8 × 3,9 kΩ		N1
1	DIL-Schalter 8fach		S1
2	20polige IC-Fassung		
2	16polige IC-Fassung		
6	14polige IC-Fassung		

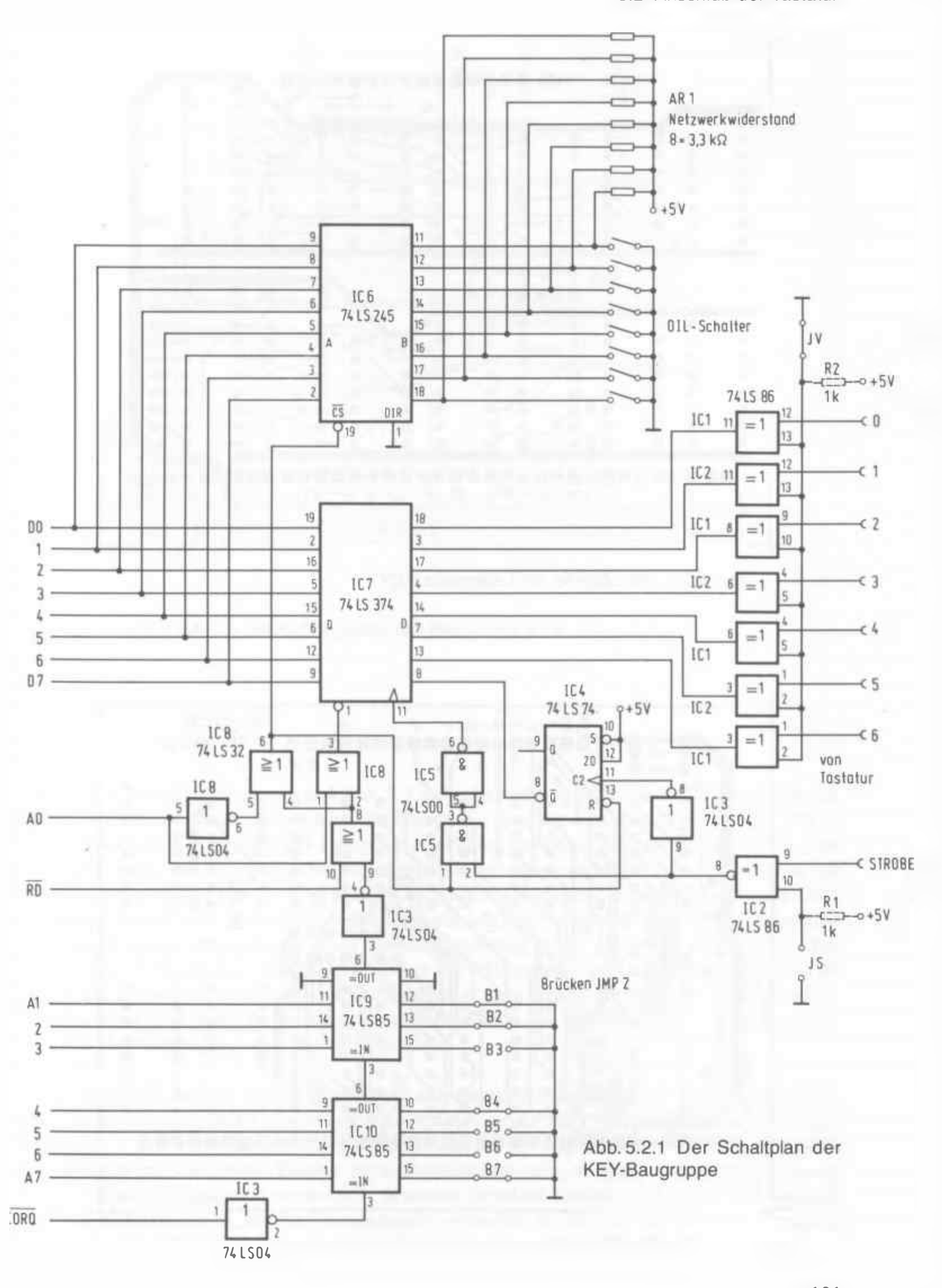


Abb. 5.2.1 Der Schaltplan der KEY-Baugruppe

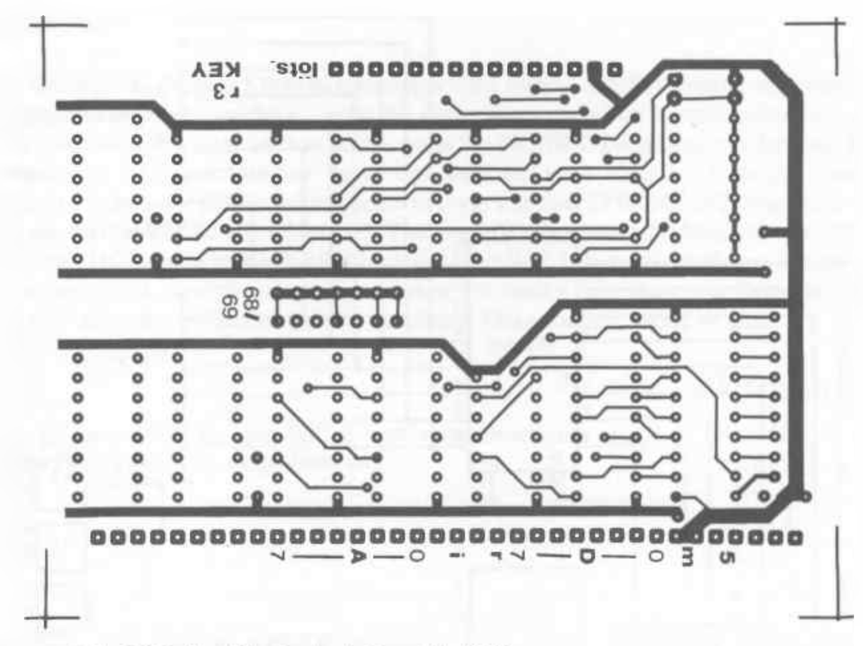


Abb. 5.2.2 Die Lötseite der Leiterplatte KEY

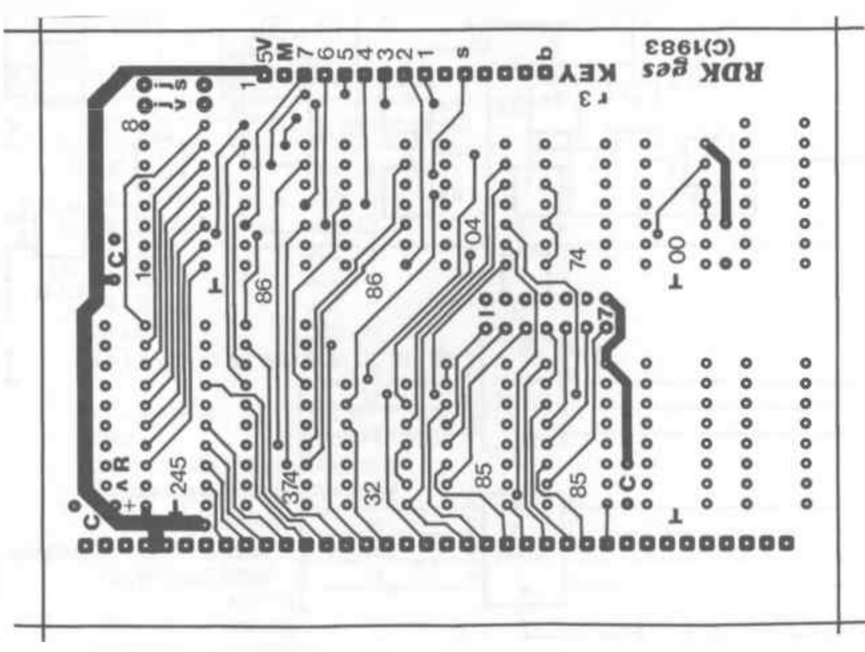


Abb. 5.2.3 Die Bestückungsseite der Leiterplatte KEY

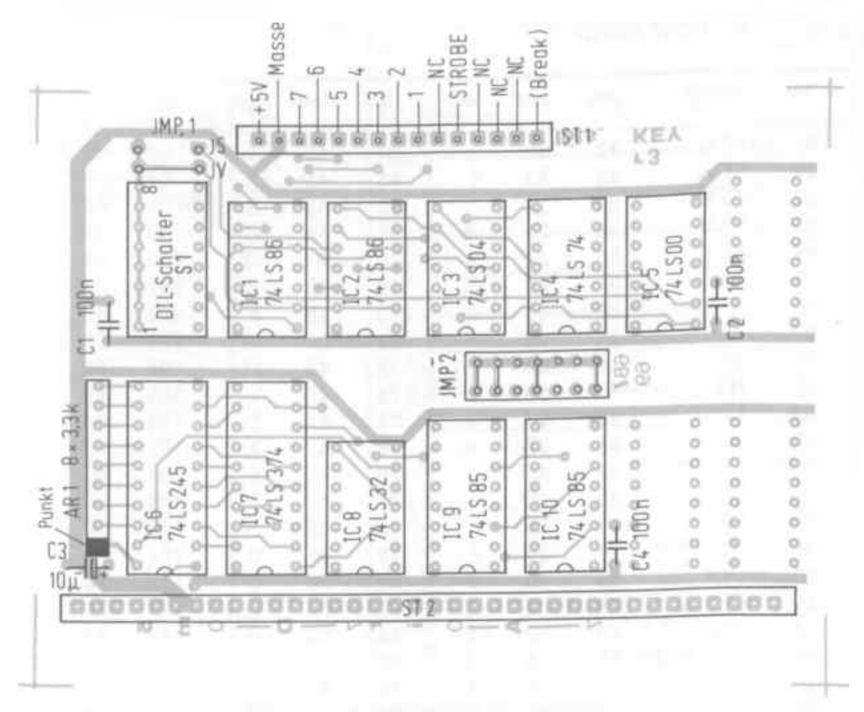


Abb. 5.2.4 Der Bestückungsplan der Baugruppe KEY

Abb. 5.2.3 die Bestückungsseite und *Abb. 5.2.4* den Bestückungsplan. Wenn Sie den Schaltplan genau betrachten, hat er ein bißchen Ähnlichkeit mit dem der IOE-Platine. Jedenfalls gibt es Adreßleitungen, mit welchen der Computer die Platine auswählt. Sie reagiert auf $\overline{IORQ}$. Der Computer kann mit $\overline{RD}$ lesen. Das Flipflop (IC4) und einige andere ICs helfen, die von der Tastatur hergestellten Bits in den Speicher 74LS374 einzubringen. Im Schaltplan ist ein DIL-Schalter eingezeichnet. Dieser DIL-Schalter besteht aus acht einzelnen Schaltern, die zusammen in einem Gehäuse untergebracht sind, das die Abmessung eines achtpoligen ICs besitzt. Man nennt diese IC-Form auch Dual-In-Line, daher die Abkürzung DIL. Dieser DIL-Schalter wird aber zunächst beim Grundprogramm nicht benötigt, er kann daher auch wahlweise entfallen. Man benötigt den DIL-Schalter für die Einstellungen bei Flomon 4.0. Seine Einstellung wird durch IC6 auf den Bus übertragen, wenn der Prozessor will.

Die Baugruppe wird nach Plan mit Fassungen versehen und die Bauelemente werden eingesetzt. Nun muß man noch die Verbindung zwischen Tastatur und KEY-Baugruppe herstellen. Dazu zeigt *Abb. 5.2.5* die Belegung der beteiligten Stecker. Mit den Brücken JMP1 auf der KEY-Platine kann man die Art des Tastaturtaktes einstellen und auch, ob positive oder negative Logik bei den Tastatur-Bits benutzt werden soll. In unserer Standard-Tastatur (Cherry) befindet sich ein eigener Mikroprozessor, der die Tasteneingabe verwaltet. Er hat die Aufgabe, die Tasten zu entprellen (siehe Folge „Geschafft, er schwingt“) und den Tasten duale Codes zuzuordnen.

Tabelle 5.2.2 Die ASCII-Tabelle

dez	hex	ASCII	dez	hex	asc	dez	hex	asc	dez	hex	asc
0	00	NUL	32	20		64	40		96	60	'
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	–	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	S1	47	2F	/	79	4F	0	111	6F	o
16	10	DLE	48	30	0	80	50	p	112	70	P
17	11	DC1 XON	49	31	1	81	51	Q	113	71	q
18	12	0C2	50	32	2	82	52	R	114	72	r
19	13	DC3 XOFF	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	798	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	|	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	–	127	7F	

ASCII ist Standard

Mit sieben Datenleitungen kann man 128 Tasten „codieren". Es gibt eine Standard-Zuordnungstabelle, die man ASCII-Tabelle nennt (American Standard Code for Information Interchange). Diese Tabelle ist nach DIN 66003 unter der Bezeichnung ISO-7-Bit-Code genormt *(Tabelle 5.2.2)*. Darin sind drei Spalten abgebildet. Einmal Dezimalcode, dann die sedezimale Codierung (HEX) und dann das ASCII-Zeichen selbst. Neben Großbuchstaben sind auch Kleinbuchstaben, Zahlen und Sonderzeichen in der Tabelle vorhanden. Der Wertebereich von 0 bis dezimal 31 umfaßt sogenannte Steuerzeichen. Sie sollen spezielle Funktionen auslösen, wie zum Beispiel „Zeilenvorschub" (LF) oder „Wagenrücklauf" (CR) usw. Wenn man die KEY-Baugruppe auf den Bus steckt und die Tastatur anschließt, so meldet sich nach dem Einschalten wieder das Grundprogramm auf dem Bildschirm. In der linken unteren Ecke blinkt ein sogenann-

Abb. 5.2.5 So wird die KEY-Platine mit der Tastatur verbunden, wenn es unsere Standard-Tastatur ist

Brucken	KEY				CHERRY Low-Cost-Tastatur
JV einsetzen JS offenlassen	+5V	1		1	+5V
	M	2		2	Masse
	7	3		3	Bit 7
	6	4		4	Bit 6
	5	5		5	Bit 5
	4	6		6	Bit 4
	3	7		7	Bit 3
	2	8		8	Bit 2
	1	9		9	Bit 1
		10	offen	10	Enable I
	S	11		11	Strobe
		12	offen	12	AKD
		13	offen	13	Enable II
		14		14	
	b	15	- - - - -	15	Break

ter Cursor. Dies ist ein helles Feld, das angibt, wo die nächste Schreibstelle liegt, wenn man Buchstaben von der Tastatur eingibt.

Experimente

1. Man drückt die Taste „A“ auf der Tastatur. Auf dem Bildschirm erscheint dann ein kleines „a“, der Cursor wird um eine Position nach rechts verschoben. *Abb. 5.2.6* zeigt den Bildschirm.
2. Wenn man erneut die Taste „A“ drückt, verschwindet der Buchstabe wieder und das Cursor-Zeichen blinkt im linken Teil des Feldes.
3. Sollte nach dem Einschalten schon ein Buchstabe auf dem Bildschirm vorhanden sein, so drückt man einfach irgendeine Taste, zum Beispiel „A“, und der Buchstabe verschwindet.
4. Groß- und Kleinumschaltung: Wenn man große Buchstaben eingeben will, so muß man wie bei der Schreibmaschine eine zusätzliche Taste drücken. Diese ist auf der Tastatur mit „SHIFT“ bezeichnet. Dabei geht man wie folgt vor. Die Taste SHIFT drücken und den Finger drauflassen. Dann mit einem anderen Finger den gewünschten Buchstaben drücken und erst nach Freigabe der Buchstabentaste auch die SHIFT-Taste wieder loslassen.

Wenn man einmal die CTRL-Taste (CONTROL-Taste) benötigt, so bedient man sie genauso als ob es eine SHIFT-Taste wäre, also vor der anderen drücken und nachher loslassen. Mit der Control-Taste kann man die Codierung verändern. Allerdings ist die Art der Änderung von Tastatur zu Tastatur verschieden. In unserem Fall ergibt sich bei den Großbuchstaben zum Beispiel immer eine (in dezimal) um 64 verminderte Code-Zahl.

```
RDK-Grundprogramm
1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter
```



Abb. 5.2.6 Das Anfangsmenü, das sich immer nach dem Einschalten und nach dem Reset zeigt. Von dort aus beginnt Ihr Weg in die Welt der Software und der Fähigkeiten unseres Computers

```
RDK-Grundprogramm
1 = aendern
2 = starten
3 = ansehen
4 = Symbole
W = weiter
```



Abb. 5.2.7 Mit w kann man das nächste Menü, das nächste Angebot an Funktionen des Computersystems, erreichen

Auf der Tastatur gibt es auch noch ein paar Tasten mit speziellen Beschriftungen. Die Taste ALPHA-LOCK ist ein Umschalter. Wenn man sie betätigt, erscheinen alle Buchstaben als Großbuchstaben. Dabei werden aber die Ziffern normal dargestellt, die beim Drücken der SHIFT-Taste ihre zweite Bedeutung bekommen würden (die immer über der Zahl abgedruckt ist). Das gleiche gilt auch für die anderen Tasten, die eine zweite Bedeutung bei SHIFT haben. Das Zeichen auf der oberen Tastenhälfte ist nur über die SHIFT-Taste erreichbar, ALPHA-LOCK schaltet allein Buchstaben auf große Darstellung um.

Deshalb gibt es auch noch eine LOCK-Taste, die wie die SHIFT-Taste wirkt, jedoch mit Feststellfunktion.

Es gibt noch eine Reihe von weiteren Steuertasten. Zum Beispiel „ESC“, „DEL“, „BREAK“, „LINE-FEED“ und „CR“. Mit der Taste „CR“ kann man dem Rechner mitteilen, wann eine Eingabezeile beendet ist. „CR“ bedeutet „Carriage Return“ oder „Wagenrücklauf“. Mit der Taste „DEL“ kann man ein versehentlich falsch eingegebenes Zeichen wieder löschen, denn „DEL“ bedeutet „Delete“ oder „Löschen“. Ähnlich verhält es sich mit „BS“, das bedeutet Back Space, oder Zeichen zurück. Die Taste „ESC“ wird oft gebraucht, um einen Programmlauf zu unterbrechen.

Dann gibt es noch eine lange Taste. Mit dieser Taste wird, wie bei der Schreibmaschine, ein Leerraum, ein „Blank“, eingegeben, um zum Beispiel Wörter voneinander trennen zu können.

Ein letzter Versuch

1. Die Taste „w“ (mit einem Großbuchstaben beschriftet) wird gedrückt. Auf dem Bildschirm erscheint *Abb. 5.2.7*.
2. Die Taste „CR“ wird gedrückt. Damit sagt man dem Rechner, daß die Eingabe beendet ist und daß man wünscht, daß der Befehl ausgeführt werde. Die Eingabe von „w“ steht für „weiter“. Gemeint ist, daß das nächste Menü ausgegeben werden soll. *Abb. 5.2.8* zeigt den neuen Bildschirminhalt.
3. Gibt man wieder „w“ und „CR“ ein, so ergibt sich *Abb. 5.2.9*.
4. Und bei nochmaliger Eingabe von „w“ und „CR“ *Abb. 5.2.10*.

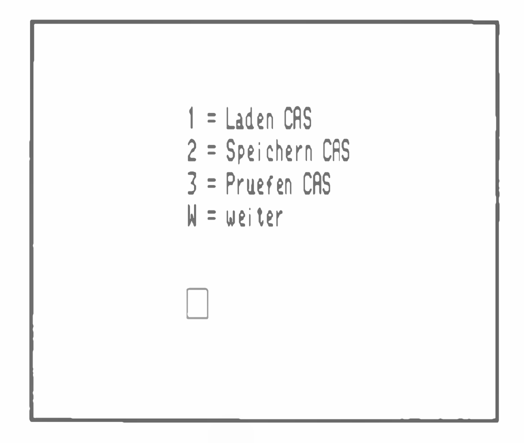


Abb. 5.2.8 Das sind alles Fähigkeiten des NDR-Klein-Computers, die Sie noch kennenlernen werden

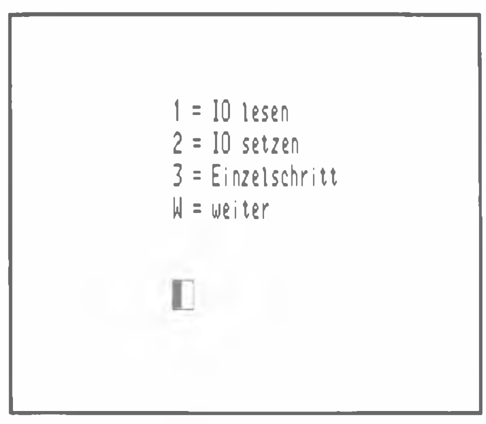


Abb. 5.2.9 Es geht noch weiter mit w

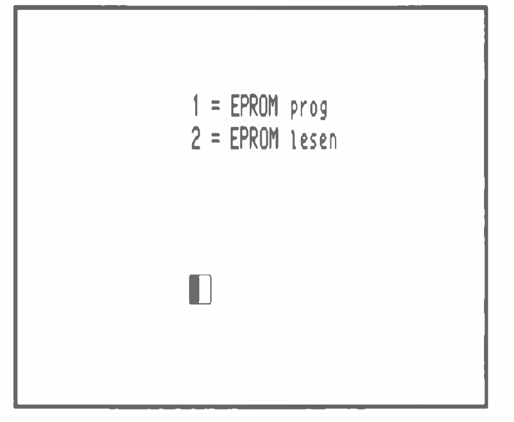




Abb. 5.2.10 Nach diesem Menü würden Sie mit w wieder auf das Anfangsmenü stoßen

Aufgaben

1. Warum muß man ein Bild in Zeilen zerlegen, wenn man es auf einem Bildschirm darstellen will?
2. Welchen Code besitzt der Buchstabe „z“ gemäß der ASCII-Tabelle? Angabe in sedezimaler und dualer Schreibweise?
3. Was bewirkt die Taste „CR“?

6 Ein Vorgeschmack von Software

6.1. Das Grundprogramm und die Schildkröte

Alles, was bisher im Buch geschah, sollte Ihnen etwas Gefühl für die Computerelektronik vermitteln. An mancher Stelle konnte nicht alles bis ins letzte erklärt werden, weil dann zu viel und auch zu spezieller Stoff dargeboten worden wäre. Vielleicht ist Ihnen der Start in die Hardware aber so gelungen, daß Sie dort jetzt auch schon alleine weiterkommen. Für alle, die eher am Programmieren interessiert sind, beginnt jetzt der Teil im Buch, der in die Software hineinführt. Hardware-Vorkenntnisse benötigt man dabei nicht.

Um einem Computer sagen zu können, was er tun soll, muß man erstens wissen, was der Computer kann und zweitens wie man ihm dann das Gewünschte befiehlt.

Unser Computer kann zum Beispiel besonders gut auf den Bilschirm zeichnen. Der amerikanische Mathematiker Seymour Papert hat für solche Computer eine Sprache entwickelt, die er Schildkrötensprache nennt. Sie besteht aus besonders einprägsamen Befehlen. Sie verwendet eine Schildkröte als Symbol, weil sie in den USA eine besonders auch Kindern vertraute Figur ist. *Abb. 6.1.1* zeigt, daß unsere Schildkröte auf dem Bildschirm durch ein Dreieck dargestellt ist.

Diese Schildkröte kann sich bewegen. Und zwar einmal vorwärts oder rückwärts. Dabei hinterläßt sie eine Schreibspur *(Abb. 6.1.2)*.

Dann kann sich die Schildkröte auch nach rechts oder links drehen *(Abb. 6.1.3)*. Wenn sie danach wieder schreitet, wird eine Spur in die neue Richtung gezeichnet, wie in *Abb. 6.1.4* sichtbar.

Auf diese Weise kann man Bilder zeichnen. *Abb. 6.1.5* zeigt ein Zehneck. Eine solche Schildkröte ist in den Computer einprogrammiert und soll einmal in Gang gesetzt werden. Dazu

Abb. 6.1.1 Ein Dreieck, das eine Schildkröte symbolisieren soll

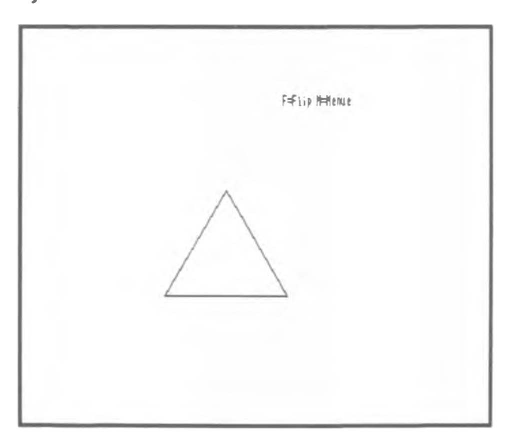
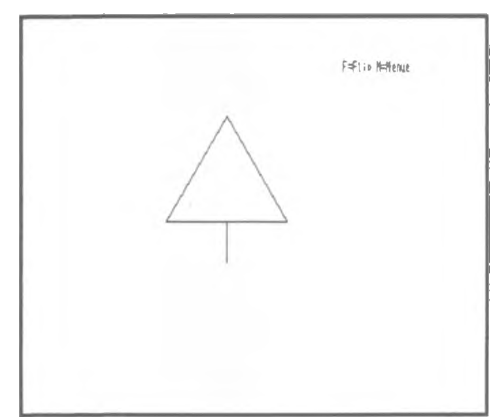


Abb. 6.1.2 Die Schildkröte hinterläßt eine Spur, wenn sie schreitet



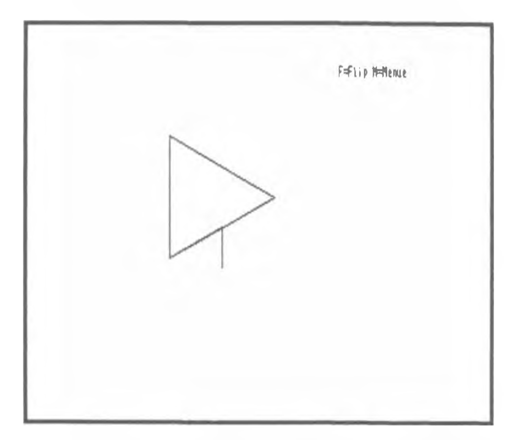


Abb. 6.1.3 Die Schildkröte kann ihre Richtung ändern

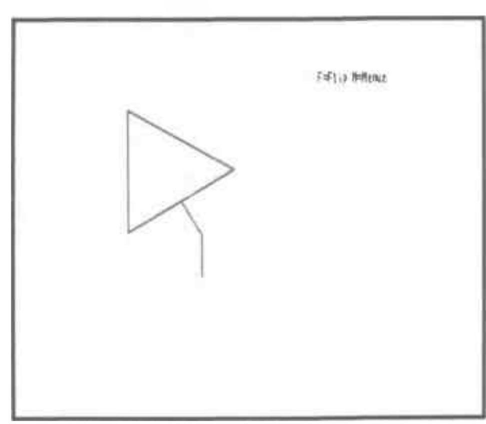


Abb. 6.1.4 Mit neuer Richtung schreiten

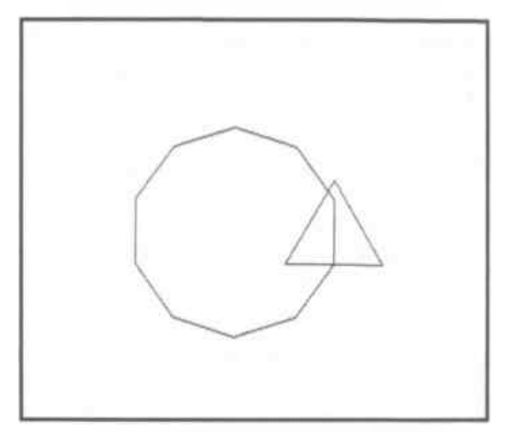


Abb. 6.1.5 Ein Zehneck ist durch Schreiten und Drehen der Richtung um einen bestimmten Winkel gekennzeichnet



Abb. 6.1.6 Das Grundmenü. Es bietet Ihnen eine Auswahl von Kommandos an, die Sie jetzt dem Computer geben können. Die zugehörigen Aktionen sind im Grundprogramm fest eingebaut. Sie werden dazu benötigt, daß Sie mit dem Computer bequem umgehen können

müssen die EPROMs mit dem Grundprogramm eingesetzt sein und dann muß der Computer eingeschaltet werden.

Er meldet sich mit dem Grundmenü *(Abb. 6.1.6)*.

Das Menü erscheint

Menü bedeutet Auswahl. Es gibt eine Auswahl von Befehlen, die dem Computer jetzt gegeben werden können.
Das Feld **„ändern“** bedeutet, daß man Programme oder Daten eingeben und verändern kann.

„starten" dient zum Starten eines Programms, nachdem es eingegeben wurde. Mit **„ansehen"** kann man sich den Inhalt eines Speicherbereichs ansehen.

Das Menü **„Symbole"** wird später zur Ausgabe von selbst definierten Namen benötigt.

Diese Befehle dienen nur dazu, den Computer bequem bedienen zu können. Sie haben mit der Schildkröte noch nichts zu tun.

Aus dem Menü benötigen wir vorerst nur die ersten beiden Befehle. Der Computer soll jetzt zum ersten Mal programmiert werden. Das geschieht mit dem Befehl „ändern", denn jede Eingabe ändert auch irgend etwas am bisherigen Zustand. Dazu wird der Buchstabe „1" eingetippt. Er erscheint dann links unten neben dem blinkenden Cursor-Feld. *Abb. 6.1.5* zeigt das Menü mit der Eingabe. Wenn vor der Eingabe schon ein Buchstabe zu sehen war, so kann man ihn mit der Taste „DEL" weglöschen und danach die gewünschte Zahl eingeben. Nun muß dem Computer gesagt werden, daß die Eingabe beendet ist. Die Taste „CR" teilt dies in dieser Situation dem Computer mit (bei manchen Tastaturen ist die CR-Taste mit einem gewinkelten Pfeil beschriftet). CR ist die Abkürzung für „Carriage Return" und heißt auf deutsch: Wagenrücklauf. Der Name wurde von der Schreibmaschinentechnik übernommen, denn dort gibt es tatsächlich einen Wagen, der zurückläuft, wenn man die Taste „CR" drückt. Wenn man die Eingabe so beendet hat, wird der Computer den angewählten Befehl ausführen. Wenn man die CR-Taste gedrückt hat, so erscheint in unserem Fall das „Änderungsmenü" auf dem Bildschirm. Dem Computer wurde also gesagt, daß er sich für das Ändern oder Neuprogrammieren bereitmachen soll, und er hat es befolgt.

Der Cursor, das ist die Marke, bei der das nächste Zeichen erscheint, wenn man eine Taste drückt, blinkt jetzt in einem Feld, das links mit „Adr:" beschriftet ist. Der Rechner wartet jetzt auf die Angabe einer Adresse.

Programme werden im Speicher abgelegt. Ein Speicher besteht aber aus vielen Speicherzellen. Damit man eine einzelne Zelle herausfinden kann, besitzt sie eine Adresse. Diese Adresse ist eine



Abb. 6.1.7 Hat man "ändern" aufgerufen, erscheint dieses Bild auf dem Sichtgerät. Der Computer wartet jetzt darauf, daß Sie ihn programmieren. Dabei muß der Programmbeginn festgelegt werden



Abb. 6.1.8 Der Bildschirm zeigt jetzt vierstellige Speicherzellen-Nummern sedezimal an. Daneben nach dem Doppelpunkt auch den Inhalt. Das rechteckige Feld kann mit Befehlen an den Computer beschrieben werden

Zahl, mit der die Speicherzellen durchnumeriert sind. Die Angabe von Speicheradressen erfolgt bei unserem Grundprogramm in sedezimaler Schreibweise. Die erste Adresse, auf der man Programme ablegen kann ist 8800. Das liegt an der Konstruktion der SBC-Karte. Man tippt also die Zahl 8800 ein und erhält *Abb. 6.1.7*. Anstelle der Zahl 8800 könnte man übrigens den fest vereinbarten Namen RAM, also die Buchstaben „R“, „A“ und „M“ eintippen. Das Grundprogramm verwendet bei Nennung von RAM automatisch den ersten Speicherplatz, der frei ist.

Jetzt wird die Taste „CR“ gedrückt, um die Eingabe zu quittieren. Wenn man aber vorher einen Tippfehler gemacht hatte, so kann man falsche Zeichen von CR mit der Taste „DEL“ löschen und die richtigen Zeichen neu tippen. Der Computer wertet die Eingabe erst dann aus, wenn man die Taste „CR“ gedrückt hat. Nach der Eingabe von „CR“ erscheint *Abb. 6.1.8*, wo es eine Vielzahl an Informationen gibt.

Zunächst die Angabe „8000:00“. Dies ist der momentane Inhalt der Speicherzelle 8800. Der Wert ist hier 0, das muß aber nicht immer so sein. Bei jedem einzelnen Exemplar unseres Computers wird etwas anderes erscheinen, denn nach dem Einschalten des Computers nehmen die Speicherzellen einen willkürlichen Wert an. Dieser Wert interessiert uns deshalb nicht.

Es wird ebenfalls noch der Inhalt der Speicherzelle 8801 ausgegeben. Auch dieser Wert ist zunächst vom Zufall eingestellt. Wir betrachten nur das breite Feld mit dem blinkenden Cursor. Dort erwartet das Grundprogramm eine Eingabe.

Im unteren Bildfeld ist eine Kurzerklärung zu sehen. Dort steht zum Beispiel „M = Menü“, gemeint ist, daß man, wenn man die Taste M drückt und mit „CR“ abschließt, wieder ins Grundmenü zurück gelangt. Oder „R = Adr“ bedeutet, wenn man die Taste „R“ drückt und dann „CR“, so kann man eine neue Adresse eingeben. Jetzt kann es ans eigentliche Programmieren gehen.

Programmieren, was ist das?

Es ist leider so, daß durch viele technische Umstände das Programmieren von Computern oft viel schwieriger erscheint, als es in Wirklichkeit sein müßte. Wie gesagt, zum Programmieren gehört ein Sack voller Fähigkeiten eines Computers und eine Benennung dieser Fähigkeiten, damit man dem Computer hintereinander aufschreiben kann, in welcher Reihenfolge er seine Fähigkeiten ausüben soll. Die Schwierigkeit ist, daß ein Computer zunächst sehr merkwürdige Fähigkeiten zu haben scheint, die einem normal denkenden Menschen oft nicht einmal als nützlich erscheinen. Sie müssen sich aber vorstellen, daß die Ingenieure, die einen Computer entworfen haben, sehr genau darüber nachgedacht haben, welche Fähigkeiten, welchen Befehlssatz sie in den Computer einbauen sollen, damit man aus diesen Befehlen dann alle nur gewünschten Aktionen des Computers zusammenbauen kann.

Unser Computer ist nun so gebaut, daß er in seinem Innersten die Befehle des Z80-Mikroprozessors besitzt, denn dieser Prozessor arbeitet in unserem Computer. Der Befehlssatz des Z80 besteht aus sehr vielen verschiedenen Befehlen, die in Zahlen zwischen Null und 255 verschlüsselt sind. Dies sind alles Zahlen, die gerade in ein Byte hinein passen, wenn man sie binär darstellt, wie es der Z80 auch verlangt. Unser Grundprogramm selbst gibt diese und auch alle anderen Binärwörter im Speicher sedezimal auf dem Bildschirm aus und akzeptiert von der Tastatur vorwiegend sedezimale Eingaben, wenn man den Speicher ändern will. Es darf Sie also nicht wundern, wenn die Befehle unseres Computers scheinbar sinnleere zweistellige Sedezimalzahlen sind. Es wird Ihnen alles Schritt für Schritt klarwerden. Außerdem sollten Sie darüber nachdenken, daß in unserem Computer ein ziemlich umfangreiches Programm arbeitet, das Grundprogramm, das es erst möglich macht, daß Sie den Computer programmieren können.



Abb. 6.1.9 So sieht der Bildschirm aus, ehe CR getastet wird

Eine Linie wird gezeichnet

Die Schildkröte soll eine Linie zeichnen. Dazu wird jetzt ein Programm eingegeben. Der erste Befehl lautet:

21 #50.W

und bedeutet: Lade den dezimalen Wert 50. Die Zahl 21 bedeutet „Lade". Sie ist der sogenannte Befehlscode, der Befehl selbst sozusagen. Der Wert „#50.W" ist die Beschreibung der Zahl, die geladen werden soll. Damit soll die Anzahl der Schritte der Schildkröte definiert werden. Das Zeichen „#" bedeutet „Dezimale Eingabe", denn der Rechner verarbeitet auch sedezimale Zahlen. Ohne das „#"-Zeichen nimmt er automatisch an, daß ihm eine sedezimale Zahl eingegeben werden soll. Der Zahlenwert ist also 50. Das „W" ist eine Angabe des zugelassenen Wertebereichs. Er muß hier angegeben werden, denn das Grundprogramm kennt auch noch andere Zahlensorten, die später erklärt werden.

Es wird also folgendes eingetippt (*Abb. 6.1.9*): Die Taste „2", dann „1", dann die lange Taste, die ein Leerzeichen erzeugt. Dann wird die Taste „#" gedrückt. Dabei muß man zuerst die SHIFT-Taste drücken und gedrückt lassen. Dann wird zusätzlich die Taste „3" gedrückt, über der sich auch das „#-Symbol befindet. Achtung: es gibt Tastaturen, bei denen das Zeichen an einer anderen Stelle liegt. Nun wird die Taste „5" betätigt, dann „0". Achtung: Die Taste „0" ist oben bei den Zifferntasten eingereiht. Man darf sie nicht mit dem „o" verwechseln, das man bei Schreibmaschinen oft als 0 benutzt. Der Computer würde das nicht verstehen. Dann wird die Taste „." gedrückt und schließlich die Taste „W". Wenn auf dem Bildschirm nun ein kleines „w" erscheint, so ist das nicht schlimm, denn das Grundprogramm versteht Groß- und Kleinschreibung. Ein großes „W" erhält man, wenn man zusätzlich die Taste SHIFT drückt.

Nun tut sich noch gar nichts, klar, denn man muß jetzt die Taste „CR" als Quittung drücken, erst dann übernimmt der Rechner die Eingabe.

Der Bildschirm sieht jetzt wie in *Abb. 6.1.10* dargestellt aus. Oben steht die Adresse 8800. Dann folgt:

8800 : 21 32 00



Abb. 6.1.10 Es zeigt sich, daß der Computer die drei durch die Eingabe erzeugten Sedezimalzahlen hintereinander abspeichert und dann eine Eingabe für die nächste freie Speicherzelle erwartet. Was in den drei Zellen 8800, 8801 und 8802 steht, ist der Befehl in Maschinencode des Z80, aber in sedezimaler Schreibweise angezeigt

aendern
Adr:8800
8800 : 21 32 00
8803 : 00
8804 : 00
8803 cd schreite
+ =Adr + 1 - =Adr - 1
M =Menue R =Adr
cr =ein Befehl weiter

Abb. 6.1.11 CD ist die Codierung eines Befehles, mit dem man komplizierte Funktionen, die vorgefertigt im Grundprogramm eingebaut sind, aufrufen kann. Hier wird die Funktion schreite aufgerufen, die die vorher eingegebene Zahl nimmt und die Schildkröte auf dem Bildschirm um diese Anzahl von Schritten weitersteuert

Der Inhalt der Speicherzelle 8800 ist 21. Dann folgt 32 und dann 00. Wo ist aber die Zahl 50 geblieben?

Das Grundprogramm hat die Zahl 50 in die sedezimale Zahl 32 umgewandelt ($3 \cdot 16 + 2 = 50$).

Woher kommt die „00" am Schluß? Sie ergibt sich, weil der Wertebereich mit .W angegeben wurde. Der Computer kann dann beim Ladebefehl eine dezimale Zahl von + 32 767 bis − 32 768 auswerten. Daraus macht er eine sedezimale Darstellung von 0 bis 7FFF und von FFFF bis 8000. Die Zahl 8000 entspricht dann der dezimalen Zahl − 32 768. Es wird die sogenannte Zweierkomplementdarstellung negativer Zahlen benutzt, was im Moment aber nicht weiter diskutiert werden soll, da die Umrechnung vom Grundprogramm automatisch durchgeführt wird. Die dezimale 50 ist also in 0032 umgewandelt worden, weil der Rechner intern führende Nullen mitnotiert. Und diese beiden führenden Nullen tauchen oben hinter der 32 auf, weil der Rechner gemeinerweise immer erst die beiden niederwertigen Stellen und dahinter erst die höherwertigen im Speicher ablegt. Diese Eigenart ist konstruktionsbedingt.

Bisher wurde dem Rechner noch nicht gesagt, was er mit der Zahl 50 tun soll, außer sie zu laden. Dazu muß ein weiterer Befehl eingegeben werden.

Man tippt „cd schreite", wie in *Abb. 6.1.11* angegeben. Dabei kann man Groß- oder Kleinbuchstaben verwenden. Man achte darauf, daß man den Befehl „cd" von „schreite" durch ein Leerzeichen (lange Taste) trennt.

Wenn man dann „CR" drückt, so erscheint *Abb. 6.1.12*. Der Code „CD" bedeutet „rufe auf". Rechts wird angegeben, was aufzurufen ist, nämlich der Befehl „schreite". Damit wird dem Rechner gesagt, er soll die Schildkröte schreiten lassen. Wie weit, das wurde durch den vorherigen Befehl, dem Lade-Wert-Befehl festgelegt.



Abb. 6.1.12 Das macht der Computer aus CD SCHREITE

aendern
Adr:8800
8803 : CD 03 00 ¦ CD SCHREITE
8806 : 00
8807 : 00
8806 c9
+ =Adr + 1 - =Adr - 1
M =Menue R =Adr
cr =ein Befehl weiter

Abb. 6.1.13 C9 ist der Befehl, der das Ende eines Programmes anzeigt

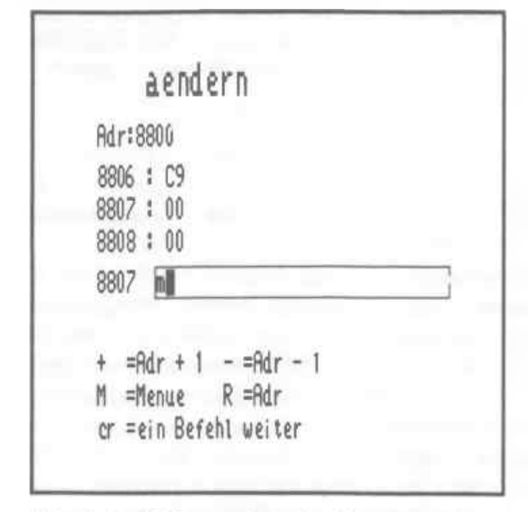


Abb. 6.1.14 Mit m geht es ins Menü zurück



Abb. 6.1.15 Wieder das Menü. Von ihm aus wird jetzt "starten" angewählt

Auf Adresse 8803 steht „CD 03 00 | CD SCHREITE“. Die linke Hälfte ist der Inhalt der Speicherzellen in sedezimaler Schreibweise, rechts, durch das Zeichen „|“ (senkrechte Linie) getrennt, steht die ursprüngliche Eingabe.

Nun muß man dem Computer noch sagen, wann das Programm zu Ende ist. Dazu gibt es den Befehl „C9“.

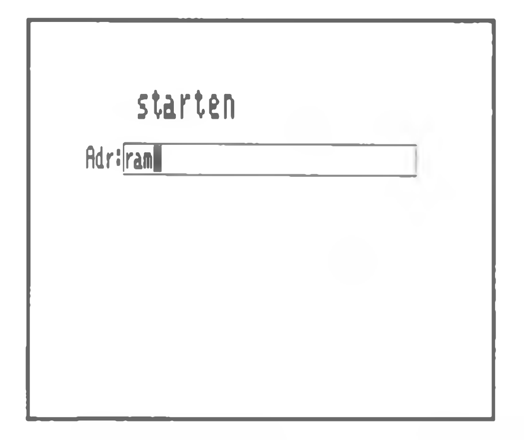


Abb. 6.1.16 Ab Adresse 8800 soll losgerechnet werden

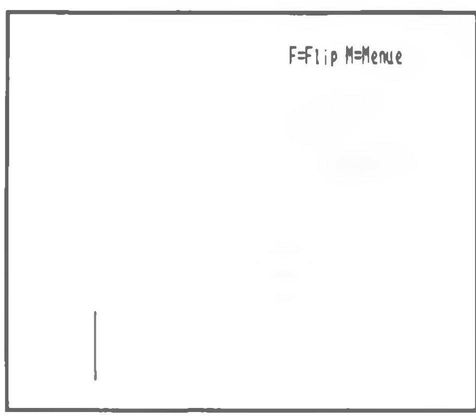


Abb. 6.1.17 Das Ergebnis eines Programmlaufes

Die Zeichen „C" und „9" werden also eingetippt. Man kann auch hier ein kleines „c" tippen, dann erscheint auf dem Bildschirm „c9". *Abb. 6.1.13* zeigt das Ergebnis. Nun noch „CR". Damit ist die Programmeingabe beendet. Nun muß man ins Menü zurück und gibt dazu den Buchstaben „M" ein. *Abb. 6.1.14* zeigt das Ergebnis.

Man gelangt aber erst dann ins Menü zurück, wenn man die Taste „CR" drückt und damit die Eingabe quittiert. Nun muß das eingegebene Programm gestartet werden. Dazu drückt man im Grundmenü die Taste „2" *(Abb. 6.1.15)* und gelangt nach Eingabe ins „Startmenü" *(Abb. 6.1.16)*. Dort muß man die Adresse eingeben, mit der das zu startende Programm beginnt. Wir müssen da 8800 angeben. Wenn man jetzt die Taste „CR" drückt, so erscheint eine Linie auf dem Bildschirm, wie in *Abb. 6.1.17*.

Das Bild zeigt nur die Linie. Auf dem Bildschirm ist in Wirklichkeit noch der Pfeil der Schildkröte dargestellt, und das Bild flimmert leicht. Man kann die Schildkröte ausblenden, wenn man die Taste „F" drückt. Wenn man die Schildkröte wieder einschalten will, so drücke man nochmals die Taste „F". Wenn man jetzt ins Menü zurück will, so gibt man den Buchstaben „M" ein, ein Tippen von „CR" kann hierbei entfallen. Man gelangt sofort wieder ins Grundmenü zurück.

Adresse 8800, die erste freie Speicherzelle, kann man auch mit dem Namen „RAM" ansprechen. Das sieht dann wie in *Abb. 6.1.18* aus.

Jetzt soll ein Quadrat gezeichnet werden. Dazu wird ein neuer Befehl benötigt. Die Schildkröte muß noch gedreht werden. Der Befehl lautet „CD DREHE". Um wieviel sich die Schildkröte dreht, hängt ebenfalls von einem Lade-Wert-Befehl ab, der davor stehen muß.
Beispiel:

```
21 #90.W
CD DREHE
```

Zuerst wird der dezimale Wert 90 geladen. Dann wird der Befehl „rufe DREHE auf" ausgeführt. Die Winkel sind in Grad anzugeben. Hier wird sich also die Schildkröte um 90 Grad gegen den Uhrzeiger drehen. Gegen den Uhrzeigersinn, weil die Mathematiker die Winkel so herum zählen.

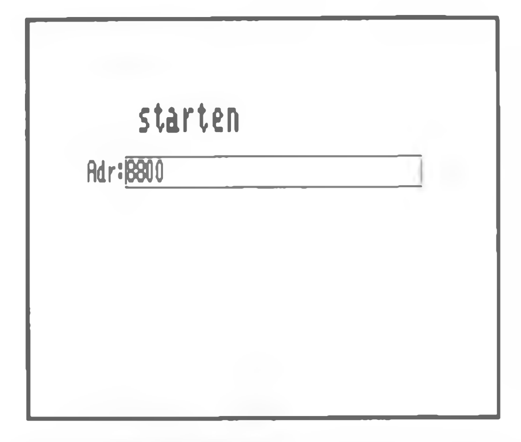


Abb. 6.1.18 Man kann den Startpunkt auch mit RAM angeben, wenn man zu Beginn des RAM-Speichers starten will

```
21 #50.W        21 #50.W
CD SCHREITE     CD SCHREITE
21 #90.W        21 #90.W
CD DREHE        CD DREHE
21 #50.W        21 #50.W
CD SCHREITE     CD SCHREITE
21 #90.W        21 #90.W
CD DREHE        CD DREHE
                C9
```

Abb. 6.1.19 Das Programm, das ein Quadrat malt

Ein Quadrat besteht aus vier Seiten. Eine Seite kann zum Beispiel durch die Folge

21 #50.W
CD SCHREITE
21 #90.W
CD DREHE

erzeugt werden. Zuerst bewegt sich die Schildkröte um 50 Punkte vorwärts, dann dreht sie sich um 90 Grad entgegen dem Uhrzeiger. Wenn man nun erneut einen Schreite-Befehl anfügt, so wird sich die Schildkröte in die neue Richtung bewegen. Wenn man also viermal die obige Sequenz aneinander fügt, so entsteht ein Quadrat. Das vollständige Programm zeigt *Abb. 6.1.19.*

Als letzter Befehl steht ein C9, dort ist das Programm zu Ende. Nun soll es eingegeben werden.

1. Dazu wird „1“ im Grundmenü angewählt („1“ tippen, dann „CR“), um das Programm einzugeben, bzw. den alten Speicherinhalt zu ändern.
2. Dann wird die Adresse eingegeben. Hier kann man wie gesagt den Namen RAM eingeben (Zeichenfolge „R“, „A“, „M“), dann „CR“.
3. Es erscheint der alte Speicherinhalt, wie *Abb. 6.1.20* zeigt, wenn man den vorherigen Versuch gemacht hat und zwischendurch den Rechner nicht abgeschaltet hat. Das bisherige Programm wollen wir weiter nutzen.
4. Wenn der Speicherinhalt dem Bild entspricht, so muß man diesen Teil des Programms also nicht neu eingeben, sondern kann durch Drücken der Taste „CR“ den nächsten freien Speicherplatz anwählen. „CR“ schaltet immer einen ganzen Befehl weiter. Man muß es zweimal tippen, dann ist man bei *Abb. 6.1.21.*
5. Auf Adresse 8806 ist der Befehl „C9“ zu sehen. Dieser Befehl wird nun durch einen neuen Befehl überschrieben. Nämlich durch „21 #90.W“. Damit ergibt sich *Abb. 6.1.22*, wenn „CR“ getippt wird.
   Auf dem Bildschirm ist dann ab Adresse 8806 die Folge „21 5A 00“ angezeigt, 5A ist der sedezimale Wert für die dezimale Zahl 90.



Abb. 6.1.20 So sieht der Bildschirm aus, wenn das erste Experiment noch nicht gelöscht ist

```
aendern
Adr:ram
882D : CD 06 00 ; CD DREHE
8830 : C9
8831 : 00
8830
+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Abb. 6.1.21 Das zeigt sich nach zwei CRs

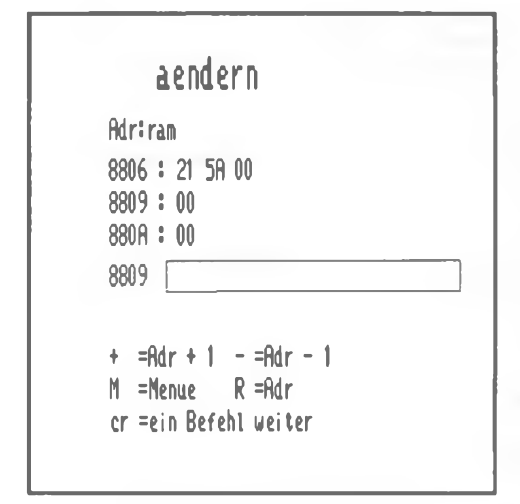


Abb. 6.1.22 So wird weiterprogrammiert, damit ein Quadrat entsteht

```
aendern
Adr:ram
882D : CD 06 00 ; CD DREHE
8830 : C9
8831 : 00
8830
+ =Adr + 1  - =Adr - 1
M =Menue    R =Adr
cr =ein Befehl weiter
```

Abb. 6.1.23 So sieht der Schlußteil des Quadratprogrammes aus

6. Nun kann der Rest des Programms eingegeben werden. Zuerst der Befehl „CD DREHE“ und so weiter bis zum „C9“-Befehl in *Abb. 6.1.19*. Wenn man den Befehl „C9“ eingegeben hat, sieht das wie in *Abb. 6.1.23* aus. Der Wert auf Adresse 8831 kann bei Ihnen auch anders aussehen, da es sich um undefiniertes Speichergebiet handelt.
7. Nun die Tasten „M“ und „CR“ drücken und man gelangt wieder ins Grundmenü.
8. Dort die Tasten „2“ und „CR“ eingeben und man gelangt ins Startmenü.

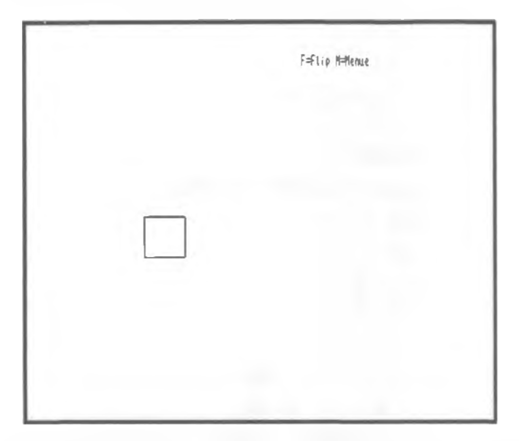


Abb. 6.1.24 Das ist das Ergebnis eines Programmlaufes

```
Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

--   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
8800 >21 32 00 CD 03 00 21 5A 00 CD 06 00 21 32 00 CD
      '  2     M        '  2     M        !  2     M
8810 03 00 21 5A 00 CD 06 00 21 32 00 CD 03 00 21 5A
           !  Z     M        !  2     M        !  Z
8820 00 CD 06 00 21 32 00 CD 03 00 21 5A 00 CD 06 00
        M        !  2     M        '  Z     M
8830 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
     I
8840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Abb. 6.1.25 Man nennt so etwas Speicherauszug. In den Zeilen stehen die Speicherinhalte in sedezimaler Darstellung. Zwischen den Zeilen sind noch die Zeichen zu sehen, die der darüberliegende Speicherinhalt ergeben würde, wenn man ihn als ASCII-Zeichen interpretiert. Nicht alle Byteinhalte ergeben ein sichtbares Zeichen auf dem Bildschirm

```
21 xxxx.W      Lade-Wert-Befehl

CD SCHREITE    Rufe-Schreite-Befehl

CD DREHE       Rufe-Drehe-Befehl

CD HEBE        Rufe-Hebe-Befehl

CD SENKE       Rufe-Senke-Befehl

C9             Ende eines Programms
```

Abb. 6.1.26 Die Liste der Befehle unserer kleinen Grafiksprache. Man kann daraus schon recht leistungsfähige Programme zusammenbauen

9. Nun die Adresse 8800 oder den Namen RAM eingeben und die Taste „CR“ drücken. Dann erscheint das Quadrat auf dem Bildschirm, wie *Abb. 6.1.24* zeigt. Wenn man die Taste „M“ drückt, so gelangt man wieder ins Grundmenü zurück.

Nun kann man sich einmal den Speicherbereich ansehen. Die Taste „3" und „CR" drücken, dann wird *Abb. 6.1.25* erzeugt. Alle Speicherzellen ab Adresse 8831 haben aber wahrscheinlich einen anderen Wert als bei uns, da diese Speicherzellen ja nicht geändert wurden und beim Spannungseinschalten ein zufälliger Wert eingestellt wurde.

Nun zwei weitere wichtige Befehle. Bisher konnte man nur geschlossene Figuren zeichnen. Es fehlte noch ein Befehl, der der Schildkröte sagt, daß sie nicht mehr zeichnen soll. Das geschieht mit dem Befehl „CD HEBE". Und wenn sie wieder fortfahren soll zu zeichnen, so kann man das mit dem Befehl „CD SENKE" erreichen.

Die vollständige Liste der benutzten Befehle zeigt *Abb. 6.1.26*.

Schreiben

Als nächstes ein paar Anregungen für weitere Figuren, die man mit dem jetzigen Befehlssatz zeichnen kann.

Sie haben sich vielleicht gefragt, wie Buchstaben auf dem Bildschirm erzeugbar sind. Hier eine Lösung mit der Zeichen-Sprache. Dazu das Programm in *Abb. 6.1.27*. Man gibt es, beginnend ab Adresse 8800 ein und startet es auch mit dieser Adresse. *Abb. 6.1.28* zeigt das Ergebnis. Die Ziffer „1" wird auf den Bildschirm gezeichnet. Im Programm gibt es eine Besonderheit. Das „—"-Zeichen. Damit ist es möglich, auch rückwärts zu schreiten oder Drehungen im Uhrzeigersinn durchzuführen. Wenn eine Zahl, wie „—#90" übersetzt wird, so ergibt sich z. B. bei "21 —#90.W" die Übersetzung „21 A6 FF". Die Zahl FFA6 ist die sedezimale Zweierkomplementdarstellung der Zahl —90. Im Speicher wird diese Zahl in zwei Hälften zerlegt und verdreht herum abgespeichert, so daß sich „A6 FF" ergibt. Die verdrehte Reihenfolge ist nötig, damit sie der Prozessor Z80 versteht. Das hat der Hersteller so bestimmt. Aber die Umrechnung und Anordnung übernimmt zum Glück das Grundprogramm, so daß man sich darum nicht kümmern muß.

```
21 -#90.W
CD DREHE
21 #20.W
CD SCHREITE
21 -#10.W
CD SCHREITE
21 #90.W
CD DREHE
21 #100.W
CD SCHREITE
21 #135.W
CD DREHE
21 #20.W
CD SCHREITE
C9
```

Abb. 6.1.27 Dies Programm malt eine 1

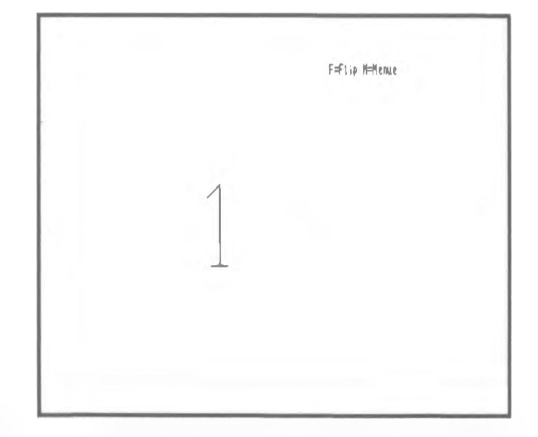


Abb. 6.1.28 Das Ergebnis des Programmes aus Abb. 6.1.27

```
21 #90.W
CD DREHE
21 #20.W
CD SCHREITE
21 -#90.W
CD DREHE
21 #200.W
CD SCHREITE
21 -#90.W
CD DREHE
21 #20.W
CD SCHREITE
21 -#90.W
CD DREHE
21 #40.W
CD SCHREITE
CD HEBE
21 #40.W
CD SCHREITE
CD SENKE
21 #40.W
CD SCHREITE
CD HEBE
21 #40.W
CD SCHREITE
CD SENKE
21 #40.W
CD SCHREITE
21 #90.W
CD DREHE
21 #20.W
CD SCHREITE
21 #90.W
CD DREHE
21 #200.W
CD SCHREITE
21 #90.W
CD DREHE
21 #20.W
CD SCHREITE
C9
```

Abb. 6.1.29 Das Programm malt eine Straße

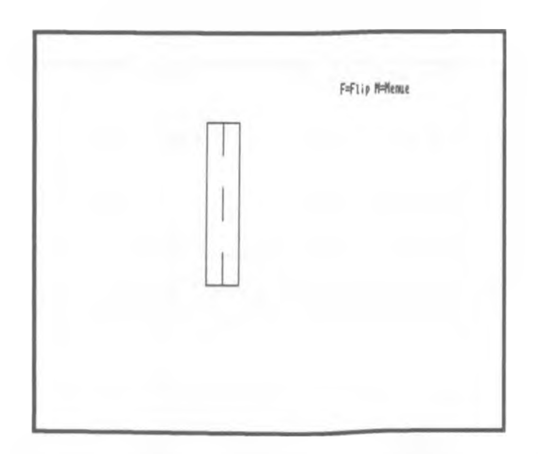


Abb. 6.1.30 Die Straße mit Mittelstrich

Speicher ansèhen

+=weiter -=rueckw R=Adr M=Menue

--	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8800	>21	5A	00	CC	06	00	21	14	00	CC	03	00	21	A6	FF	CD
8810	06	00	21	C8	00	CD	03	00	21	A6	FF	CD	06	00	21	14
8820	00	CC	03	00	21	A6	FF	CD	06	00	21	28	00	CC	03	00
8830	CC	09	00	21	28	00	CD	03	00	CD	0C	00	00	21	28	00
8840	CC	03	00	CD	09	00	21	28	00	CD	03	00	CC	0C	00	21
8850	28	00	CC	03	00	21	5A	00	CD	06	00	21	14	00	CD	03
8860	00	21	5A	00	CC	06	00	21	C6	00	CC	03	00	21	5A	00
8870	CC	06	00	21	14	00	CD	03	00	C9	00	00	00	00	00	00

Abb. 6.1.31 So sieht der Speicherauszug aus, wenn das Programm aus Abb. 6.1.27 eingetippt ist

Eine Straße wird gezeichnet

Nun ein weiteres Beispiel. Eine kleine Straße soll gezeichnet werden. Dazu das Programm in *Abb. 6.1.29*. Wenn man das Programm eingibt und startet, so ergibt sich *Abb. 6.1.30*. In *Abb. 6.1.31* ist der Speicherinhalt einmal vollständig zum Vergleich abgebildet. Dabei ist die Adresse 8879 mit dem Inhalt „C9" die letzte definierte Speicherzelle. In Ihrem Computer können danach andere Werte stehen, das stört aber nicht weiter.

Unter jedem sedezimalen Wert steht ein ASCII-Zeichen. Dies ist wie schon gesagt die Interpretation des Grundprogramms, wenn man den sedezimalen Code, der darüber steht, als ASCII-Code ansieht (siehe Folge „Schreiben lernen") und auf den Bereich 0 bis 7F abbildet. Dabei wird das Bit 7 ignoriert, denn zum Beispiel ist CD kein ASCII-Element. CD ist dual 11001101. Wenn man aber die erste Stelle wegläßt ergibt sich 1001101. Dies ist sedezimal dargestellt 4D und 4D entspricht in ASCII dem Zeichen „M". Diese Darstellung dient nur der zusätzlichen Information. Sie wird erst später gebraucht.

Übrigens besitzt der Bildschirm eine Auflösung von 512 × 256 Punkten. Für die Zeichensprache wird der Bereich auf 512 × 512 Punkte erweitert, um symmetrische Darstellungen zu ermöglichen. Die Schildkröte beginnt bei Start des Programms immer in der Mitte des Bildschirms zu zeichnen. Sie zeigt dabei in Richtung oberer Bildrand.

Aufgaben

1. Es soll ein Dreieck gezeichnet werden. Wie sieht das Programm aus?
2. Man versuche eine einfache Haus-Darstellung auf den Bildschirm zu bringen.
3. Was geschieht, wenn man die Befehle „21 #30.W" und „CD DREHE" vor das Quadratprogramm stellt? Also Programm nochmals eingeben und dabei mit der neuen Sequenz beginnen und dann das restliche Programm eingeben.
4. Ein Sechseck soll gezeichnet werden.
5. Was passiert, wenn man zu lange Linien zeichnet? Dazu versuche man folgendes Programm „21 #400.W", „CD SCHREITE", „21 #170.W", CD DREHE", „21 #400.W" und „CD SCHREITE" sowie „C9".

6.2 Blumen mit Schleife

Der Sinn des letzten Kapitels war, daß Sie feststellen sollten, daß mit ganz wenig Befehlen, den Grafikbefehlen unserer kleinen Grafiksprache, schon sehr sinnvolle Programme geschrieben werden konnten. In diesem Kapitel werden nun Befehle geschildert, die es erlauben, einmal durchprogrammierte Programmteile immer wieder zu benutzen. Das gibt der Sprache eine kräftige Struktur, mit der man leicht umgehen kann und die es erlaubt, auch mächtige Programme sicher zu beherrschen.

Es soll ein Kreis auf dem Bildschirm gezeichnet werden. Das geht mit unseren Mitteln sehr einfach. Die Schildkröte kann als kleinsten Schritt um einen Bildpunkt fortschreiten, und der kleinste Winkel, um den sie sich drehen kann, beträgt 1 Grad.

Das hat seinen guten Grund. Denn ein Bild auf dem Bildschirm besteht immer aus einzelnen Punkten. Damit kann man zwar keinen exakten Kreis darstellen, die Kreisform läßt sich aber ganz gut annähern.

Was würde geschehen, wenn man die Schildkröte um einen Schritt schreiten ließe und dann um ein Grad drehen würde, dann wieder einen Schritt schreiten, und dann ein Grad drehen usw. . . .? Es entstünde eine Art Kreis. Wann würde die Schildkröte wieder den ursprünglichen Ort erreichen? Wenn sie einen Winkel von 360 Grad durchlaufen hätte, denn dann hätte sie sich einmal um sich selbst gedreht und würde wieder in die Anfangsrichtung zeigen.

Es würde also ein 360-Eck gezeichnet worden sein.

Wenn ein solches Programm eingegeben werden soll, so hat das einen Haken, denn die Sequenz

```
21 #1.W
CD SCHREITE
21 #1.W
CD DREHE
```

muß offenbar 360mal eingegeben werden. Das sind über tausend Befehle. Dabei müssen immer wieder die gleichen Elemente niedergeschrieben werden. Bei richtigen Computern gibt es dazu ein Hilfsmittel, die Schleife (auch LOOP genannt), die ständige Wiederholungen ein und desselben Programmstücks unterstützt, ohne daß man das Stück immer wieder niederschreiben muß. Auch in unserem Grundprogramm gibt es eine derartige Hilfe. Die beiden Befehle:

CD SCHLEIFE

und

CD ENDSCHLEIFE

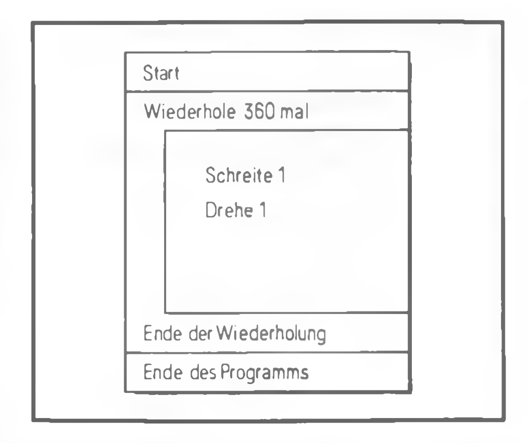
Mit Schleifen gezielt wiederholen

Mit „CD SCHLEIFE“ wird der Beginn einer Wiederholung markiert. Mit „CD ENDSCHLEIFE“ wird das Ende markiert. Alle zwischen diesen beiden Befehlen stehenden Anweisungen werden wiederholt. Wie oft? Dazu muß ein Lade-Befehl vorangestellt werden, dessen Lade-Wert die Anzahl der Durchläufe angibt. Das vollständige Programm ist in *Abb. 6.2.1* abgedruckt. Das Programm kann auch als Diagramm dargestellt werden, man nennt die Darstellungsform in *Abb. 6.2.2* Struktogramm.

```
21 #360.W
CD SCHLEIFE
21 1.W
CD SCHREITE
21 1.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Abb. 6.2.1 Dies Programm zeichnet einen Kreis, probieren Sie es aus

Abb. 6.2.2 Das Programm aus Abb. 6.2.1 als Struktogramm. Das bedeutet, daß in den Rechtecken Handlungen des Computers mit Worten niedergeschrieben sind und zwar in der Reihenfolge, in der der Computer sie ausführen soll. Besteht so eine Handlung aus mehreren Einzelhandlungen, dann kann man diese in Unter-Kästen notieren



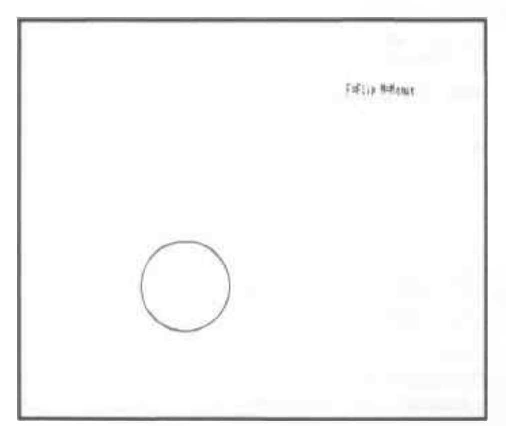


Abb. 6.2.3 Das Ergebnis eines Programmlaufes

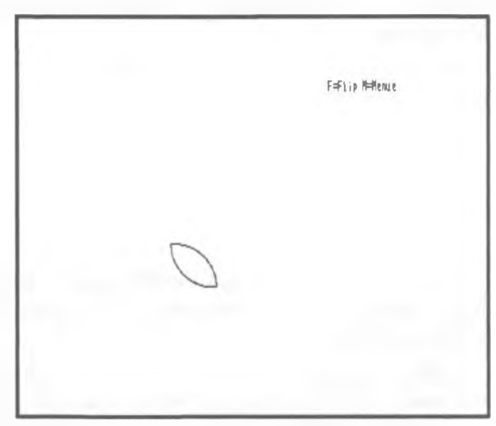


Abb. 6.2.4 Ein Blütenblatt, hier aus zwei Viertelkreisen zusammengesetzt

Nun zur Eingabe des Programms. Es wird ab Adresse 8800 eingegeben. Wichtig! Nicht das „C9“ am Schluß vergessen. *Abb. 6.2.3* zeigt das Ergebnis des Programms auf dem Bildschirm.

Aufgaben

1. Was passiert, wenn man in der Schleife um 2 schreitet?
2. Was passiert, wenn man die Schildkröte in der Schleife um je 2 Grad dreht?
3. Wie kann man kleinere Kreise zeichnen?

Eine Blume aus Teilen zusammensetzen

Es soll eine Blume gezeichnet werden. Diese Blume soll aus Blüten und einem Stengel bestehen. Das Kreisprogramm kann man dazu hervorragend verwenden. Zunächst soll ein Blütenblatt gezeichnet werden. Dieses Blütenblatt setzen wir aus zwei Viertelkreisen zusammen. *Abb. 6.2.4* zeigt ein Schema.

```
VIERTELKREIS:=$
21 #90.W
CD SCHLEIFE
21 1.W
CD SCHREITE
21 1.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Abb. 6.2.5 So zeichnet Ihr Computer einen Viertelkreis



Abb. 6.2.6 Mit dieser Anweisung wird ein Name mit dem Computer vereinbart. Er weiß danach, daß die Adresse 8800 jetzt Viertelkreis heißt

Also gilt es, zuerst einen Viertelkreis zeichnen zu können. Wie das geht, müßte aber klar sein. Man zeichnet einfach den vierten Teil eines Kreises, verwendet also nur 90 Wiederholungen anstelle der 360.

Abb. 6.2.5 zeigt das Viertelkreisprogramm. Dort ist noch eine Besonderheit zu sehen, nämlich die erste Zeile. Wenn man diese mit eintippt, so bekommt das Programm einen Namen und man kann später einfach den Namen anstelle der Adresse verwenden. Die Eingabe des Namens zeigt *Abb. 6.2.6*. Wenn man anschließend die Taste „CR“ drückt, verschwindet die Eingabe wieder, aber das Grundprogramm hat sich den Namen gemerkt. Nun kann das restliche Programm eingegeben werden. Wenn man den vorherigen Versuch durchgeführt hat, so braucht man nur die Zeile „21 #90.W“ neu eingeben, der Rest ist ja identisch mit dem Kreisprogramm.

Nun geht man zurück ins Grundprogramm und ruft den Befehl „2“ auf, also „Starten des Programms“. Dann erscheint die Meldung „ADR.:“ und jetzt kann man den Namen „VIERTELKREIS“ eingeben. Nach dem „CR“ erscheint der Viertelkreis auf dem Bildschirm.

Die Unterprogrammtechnik: Ein mächtiges Werkzeug

Wir wollen ein Blatt zeichnen und nicht nur einen Viertelkreis. Das Programm muß also noch erweitert werden. Dazu verwenden wir die sogenannte Unterprogrammtechnik. In unserem System können wir das Programm „Viertelkreis“ als neues Sprachelement verwenden und rufen es einfach mit dem Befehl „CD“ auf.

Für ein Blatt benötigt man zwei Viertelkreise, die aneinander gesetzt werden müssen. *Abb. 6.2.7* zeigt die Lösung. Dabei wird zunächst ein neuer Name definiert, nämlich „BLATT“. Unter dieser Bezeichnung läßt sich das Programm später aufrufen. Man muß sich also keine Adresse merken. Dann wird mit „CD VIERTELKREIS“ unser vorhergehendes Programm aufgerufen, so als ob VIERTELKREIS ein eigener Befehl sei. Danach muß die Schildkröte um 90

```
BLATT:=$
CD VIERTELKREIS
21 #90.W
CD DREHE
CD VIERTELKREIS
21 #90.W
CD DREHE
C9
```

Abb. 6.2.7 Dieses Programm benutzt Viertelkreis als Unterprogramm. Viertelkreis ist damit in den Befehlssatz der Grafik-Sprache aufgenommen



Abb. 6.2.8 Der Name von Blatt wird vereinbart

Grad gedreht werden, ehe der zweite Viertelkreis angeschlossen werden kann. Und dann wird erneut das Unterprogramm „VIERTELKREIS" aufgerufen. Zum Schluß wird nochmals um 90 Grad gedreht. Dies geschieht, damit die Schildkröte wieder in die Ausgangsrichtung blickt, denn sie hat dann insgesamt eine Drehung von 360 Grad durchlaufen. Damit wird das weitere Arbeiten einfacher. Man sollte bei eigenen Programmen immer darauf achten, bei geschlossenen Figuren die Schildkröte auch insgesamt um 360 Grad oder ein Vielfaches davon zu drehen.

Die Blume wird eingegeben

Das alte Programm „VIERTELKREIS" darf natürlich nicht zerstört werden und daher gehe man wie folgt vor:

1. Änderungsmenü aufrufen.
2. Eingabe der Adresse 8800 oder des Namens „VIERTELKREIS" (große und kleine Buchstaben sind zugelassen).
3. Das alte Programm wird jetzt sichtbar.

Nun muß man die Taste „CR" so oft drücken, bis der letzte Befehl des schon geschriebenen Programms sichtbar wird, also der Befehl „C9". Dann drücke man noch zweimal die Taste „CR", bis der Befehl „C9" ganz oben im Bild steht. Damit hat man einen Speicherplatz angewählt, der noch nicht durch ein Programm belegt ist. Dort kann man jetzt das neue Programm eingeben. Es beginnt mit der Definition des Namens, also mit „BLATT: = $" und sieht dann wie *Abb. 6.2.8* aus. Danach drückt man die Taste „CR", und die Eingabe verschwindet wieder, das Grundprogramm hat die Eingabe angenommen. Achtung! Am restlichen Bild ändert sich bei Namensdefinitionen nichts. (Das Zeichen „$" ist übrigens die Abkürzung für die links neben dem Kasten stehende Adresse, man könnte auch schreiben: „BLATT:=8816", das aber nur in diesem Fall, denn der Wert der Adresse ist von der jeweiligen Lage des Programms abhängig.) Die Eingabe mit dem



Abb. 6.2.9 Sie können jetzt sicher schon selbständig weiterprogrammieren

```
BLUETE:=$
21 #5.W
CD SCHLEIFE
CD BLATT
21 #72.W
CD DREHE
CD ENDSCHLEIFE
C9
```

Abb. 6.2.10 Dies Programm malt eine Blüte, wenn es gleichzeitig mit Viertelkreis und Blatt im Speicher steht. Es benutzt nämlich diese beiden Unterprogramme

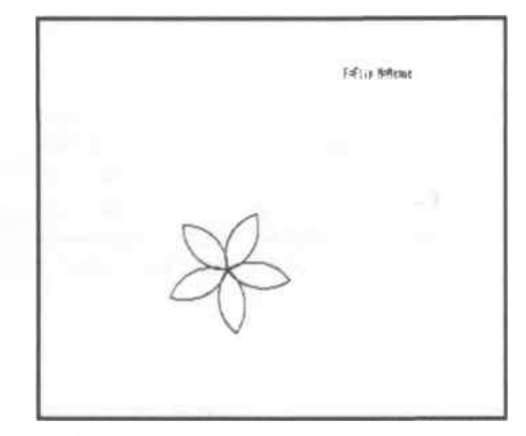


Abb. 6.2.11 Das ist die Blüte

Dollar-Zeichen ist sehr bequem. Als nächstes folgt also „CD VIERTELKREIS", dann „21 #90.W" usw. *Abb. 6.2.9* zeigt einen Ausschnitt.

Wenn das Programm gestartet wird, und diesmal mit dem Namen „BLATT", so ergibt sich wirklich ein Blatt. Hier zeigt sich der Vorteil der Namensgebung. Wer hätte noch gewußt, daß die Startadresse des Programmes Blatt 8816 ist? Die Eingabe des Namens „BLATT" ist da viel einfacher.

Nun werden die Blätter zu einer Blüte zusammengesetzt. Dazu wird die Form Blatt einfach 5 mal ausgegeben, wobei jedesmal eine Drehung von 72 Grad durchgeführt wird. Da $5 \cdot 72 = 360$ ist, ergibt sich eine geschlossene Figur. *Abb. 6.2.10* zeigt das entsprechende Programm und *Abb. 6.2.10* das Ergebnis bei Aufruf des Programms „BLUETE". Zur Programmeingabe verfährt man wie im vorherigen Beispiel. Zuerst wird das Ende des bisher eingegebenen Programmsystems gesendet. Dabei kann man bei der Adresse „BLATT" anfangen zu suchen bis „C9" im Bild oben erscheint. Dann kann das neue Programm eingegeben werden.

Wenn jetzt noch ein Stengel und zwei Blätter angefügt werden, ist die Blume fertig. *Abb. 6.2.12* zeigt das Programm „BLUME“ und *Abb. 6.2.13* das Ergebnis auf dem Bildschirm.
In *Abb. 6.2.14* ist der Speicherauszug abgedruckt. Damit kann man die Eingaben überprüfen.

```
BLUME:=$
CD BLUETE
21 -#150.W
CD SCHREITE
CD BLATT
21 -#10.W
CD SCHREITE
21 -#90.W
CD DREHE
CD BLATT
21 #90.W
CD DREHE
21 -#40.W
CD SCHREITE
C9
```

Abb. 6.2.12 Das Programm Blume



Abb. 6.2.13 Die Blume



Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

--	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8800	21	5A	00	CD	0F	00	21	01	00	CD	03	00	21	01	00	CD
8810	06	00	CD	12	00	C9	CD	00	88	21	5A	00	CD	06	00	CD
8820	00	88	21	5A	00	CD	06	00	C9	21	05	00	CD	0F	00	CD
8830	16	88	21	40	00	CD	06	00	CD	12	00	C9	CD	29	88	21
8840	6A	FF	CD	03	00	CD	16	88	21	F6	FF	CD	03	00	21	A6
8850	FF	CD	06	00	CD	16	88	21	5A	00	CD	06	00	21	D8	FF
8860	CD	03	00	C9	00	00	00	00	00	00	00	00	00	00	00	00
8870	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Abb. 6.2.14 Der Speicherauszug von Blume



Abb. 6.2.15 Das sind die bisher vereinbarten Symbole und die zugehörigen Adressen

Ein neuer Menüpunkt

Bisher hatten wir den Menüpunkt „4 = Symbole“ noch nicht verwendet. Das soll sich jetzt ändern. Wenn es aufgerufen wird, erscheint *Abb. 6.2.15*. Dort sind alle definierten Namen eingetragen. Dabei steht links neben dem Namen der Wert, dem er zugeordnet wurde. Namen sind also bei uns nur eine andere Schreibweise für Zahlenwerte.

Und so arbeitet auch die Zuordnung der Namen „SCHREITE“, „DREHE“. Auch diese Namen stehen nur anstelle von Zahlenwerten. Man könnte ebensogut die entsprechenden Zahlenwerte eingeben. Doch Namen haben hier einen Vorteil. Das Grundprogramm kann sie überprüfen. Wird ein Name eingegeben, der noch nicht definiert war, so erkennt es das Grundprogramm. Bei einer Befehlseingabe im Änderungsmenü bleibt die Adresse stehen und zeigt einem damit, daß ein Fehler gemacht wurde. Wenn man einen nicht definierten Namen als Adresse angibt, so wird z. B. beim Startmenü gar kein Programm ausgeführt, es meldet sich gleich das Grundprogramm.

Wenn man andererseits eine falsche Zahl als Adresse irrtümlich eingibt, so kann das Grundprogramm dies nicht kontrollieren – und das Programm „hängt sich“ unter Umständen auf. Dann meldet sich das Grundprogramm nie wieder. Das eingegebene Programm wird dann sogar manchmal zerstört. Also: Namen verwenden spart Arbeit.

Fehler, Fehler!

Fehlersuche ist das wichtigste beim Programmieren, denn zunächst sind in einem längeren Programm immer irgendwelche Fehler, die man finden und beseitigen muß. Dazu benötigt man den Menüpunkt „Einzelschritt“. Um dort hinzugelangen, betätigt man die Taste „W“ im Grundmenü und dann die Taste „CR“, und nochmals „W“ und „CR“. Dann erscheint *Abb. 6.2.16*. Man wählt „Einzelschritt“ durch die Taste „3“ gefolgt von einem „CR“.

Hier wird nun genau wie beim Startmenü die Adresse angefragt. Man gibt z. B. „BLUME“ ein. *Abb. 6.2.17* zeigt die Eingabe. Nach Eingabe von „CR“ erscheint *Abb. 2.6.18*. In der unteren Bildhälfte erscheint eine Fülle von Informationen, die aber zunächst nicht alle von Bedeutung sind. Rechts unten erscheint der erste Befehl, der ausgeführt werden soll. In diesem Fall „CD BLUETE“, denn das ist der erste Befehl im Programm „BLUME“. Will man, daß der Computer

1 = IO lesen
2 = IO setzen
3 = Einzelschritt
W = weiter

Abb. 6.2.16 Dieses Menü enthält den Punkt Einzelschritt



Abb. 6.2.17 Die Blume soll untersucht werden

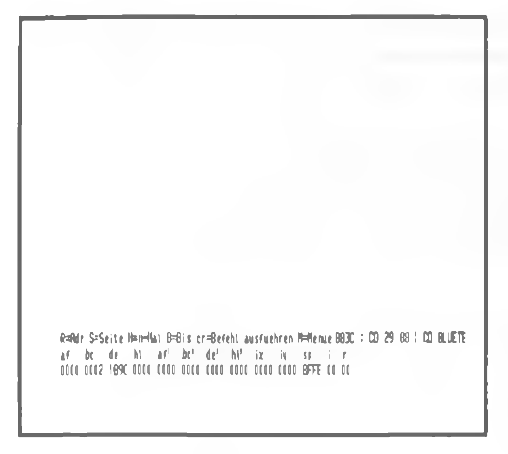


Abb. 6.2.18 Vor dem ersten Schritt

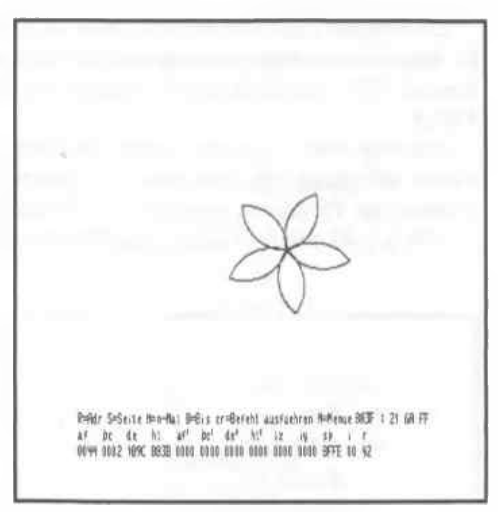


Abb. 6.2.19 Und danach

diesen Befehl ausführt, so drücke man die Taste „CR". Es ergibt sich *Abb. 6.2.19*. Der nächste Befehl, der ausgeführt wird, ist „21 6A FF", also ein Lade-Befehl. Wenn man nun „CR" drückt, wird er ausgeführt, und es erscheint der nächste Befehl.

So kann man Befehl für Befehl schrittweise ausführen und damit Fehlern auf die Spur kommen.

Der Befehl „CD BLUETE" besteht selbst wieder aus einzelnen Befehlen. Diese werden aber nicht schrittweise durchlaufen, da es einen Namen dafür gibt. Will man „BLUETE" testen, so muß man „BLUETE" selbst als Startadresse anwählen. Hat man einen Fehler gefunden, so muß man die entsprechende Stelle mit dem Änderungsmenü verändern und das Programm von da an ggf. neu eingeben.

Bei Programmen, die man selbst entwickelt, empfiehlt es sich daher, sie erstens sofort Unterprogramm für Unterprogramm auszutesten und zweitens Lücken zwischen den Programmen zu lassen, um dort später weitere Befehle unterbringen zu können, falls man welche vergessen haben sollte. Man kann dazu zum Beispiel den NOP-Befehl verwenden, dessen Code 00 lautet. Er wird einfach zwischen Programmelementen passend eingestreut.

Beispiel:

```
21 #90.W
CD DREHE
00 00 00 00 00
21 #1.W
CD SCHREITE
```

Ein neuer Befehl,

CD SCHR16TEL, damit ist es möglich um einen 1/16-Punkt vorwärts zu schreiten. Diesen Befehl benötigt man zum Beispiel, wenn man irgendwelche Kreisradien erzeugen will.

Ein Beispiel zeigt *Abb. 6.2.20*. Dabei ist im Programm noch eine Besonderheit untergebracht. Es sind diesmal zwei Schleifen-Befehle ineinandergeschachtelt. *Abb. 6.2.21* zeigt das Struktogramm. Die innere Schleife erzeugt ein 36-Eck. Beides ergibt zusammen die Figur in *Abb. 6.2.22*.

Jetzt eine letzte Aufgabe. Es soll ein Zahnrad gezeichnet werden. Dazu kann man zuerst eine Zacke definieren, die man dann zu einem Zahnrad ausbauen kann. *Abb. 6.2.23* zeigt das vollständige Programm und *Abb. 6.2.24* das Ergebnis auf dem Bildschirm.

Abb. 6.2.25 zeigt die Liste von Befehlen, die bis hierher verwendet wurden.

```
KREISE:=$
21 #36.W
CD SCHLEIFE
21 #360.W
CD SCHLEIFE
21 #3.W
CD SCHR16TEL
21 #1.W
CD DREHE
CD ENDSCHLEIFE
21 #20.W
CD SCHREITE
21 #10.W
CD DREHE
CD ENDSCHLEIFE
C?
```

Abb. 6.2.20 Das Programm Kreise mit neuem Befehl und verschachtelten Schleifen



Abb. 6.2.21 Das Struktogramm zu Kreise



Abb. 6.2.22 Das sind 36 Kreise in einem 36-Eck

Abb. 6.2.23 Zahnrad benutzt Zacke

```
ZACKE:=$
21 -#60.W
CD DREHE
21 #10.W
CD SCHREITE
21 #120.W
CD DREHE
21 #10.W
CD SCHREITE
21 -#60.W
CD DREHE
C9

ZAHNRAD:=$
21 #72.W
CD SCHLEIFE
CD ZACKE
21 #5.W
CD DREHE
CD ENDSCHLEIFE
C9
```

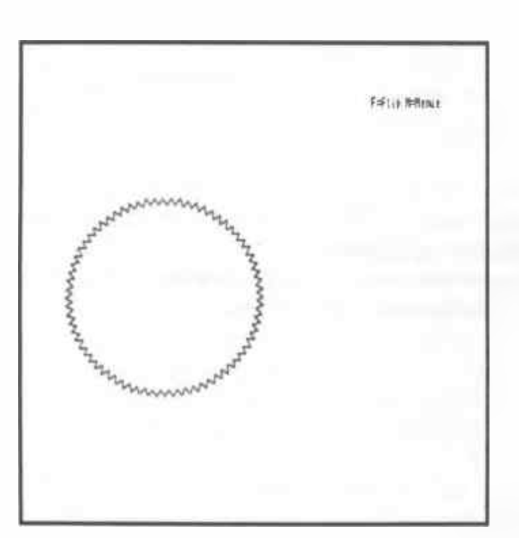


Abb. 6.2.24 Ein Zahnrad vom Computer gezeichnet

```
21 xxxx.W

CD SCHREITE

CD DREHE

CD SCHLEIFE

CD ENDSCHLEIFE

CD SCHR16TEL

name:=$

name:=xxxx


xxxx kann sein "#zahl" oder "-#zahl" oder "sedzahl"

     oder "name".
```

Abb. 6.2.25 Mit diesen wenigen Befehlen sind alle Zeichnungen angefertigt

Aufgaben

1. Was passiert, wenn man einen „CD SCHLEIFE“- oder „CD ENDSCHLEIFE“-Befehl vergißt?
2. Wie kann man das Quadrat-Programm jetzt kürzer schreiben?
3. Verschiedene Figuren sollen zunächst auf Papier gezeichnet werden und dann ein Programm dazu entwickelt werden. Beispiel: Dreiecke im Kreis angeordnet, Haus usw.

7 Peripherie

Nachdem Sie nun schon einen kleinen Eindruck von der Programmierung gewonnen haben, wollen wir an dieser Stelle weitere Hardware-Schaltungen aufbauen. Sie können sich dabei aus dem nachfolgenden Abschnitt die für Sie interessanten Teile herauspicken. Man muß keinesfalls alles aufbauen und auch nicht der Reihe nach.

7.1 Die IOE-Baugruppe, eine Universalkarte

Wenn man eigene Schaltungen entwickeln will, so ist eine Universalkarte ganz interessant. Sie besitzt ein großes Lochrasterfeld, und auf diesem kann man Schaltungen aufbauen, so um zum Beispiel Motoren zu steuern, Lampen ein- und auszuschalten oder um Schalter und Tasten abzufragen.

Damit lassen sich dann Steuerungen aufbauen, wie zum Beispiel eine Alarmanlage. Oder man entwickelt selbst einmal Schaltungen für den Computer und probiert neue ICs aus.

Auf der Universalkarte befinden sich aber auch eine ganze Reihe von ICs, die den Betrieb mit dem Computer leichtmachen, z. B. eine Dekodierschaltung und Puffer, die den Verdrahtungsteil vom Computer-Bus isolieren.

Abb. 7.1.1 zeigt die Schaltung. IC2 und IC3 sind zwei bidirektionale Bustreiber. Sie sind hier aber als Eingabeeinheit verschaltet. Der DIR-Eingang führt an den -RD-Anschluß und schaltet die Richtung um. Liegt dort ein Low-Signal an, so schalten die Treiber von B nach A, wenn auch der Eingang -CS ein Low-Signal führt.

-CS ist mit einem Dekoder (IC6) verbunden. An den Dekoder führen drei Signale, A0, A1 und über Pin 1 ein Auswahlsignal. Dieses Signal liegt genau dann auf Low, wenn ein Lesevorgang stattfindet (Pin 13, IC8) und am Ausgang von IC1, Pin 12 ein Low-Signal liegt. Das ist aber nur dann der Fall, wenn am Vergleicher IC7 alle Eingänge der A-Seite das gleiche Signal, wie auf der B-Seite (bei JMP1) haben und der Eingang am IC7, Pin 3 ein High-Signal führt. Der Eingang führt immer dann ein High-Signal, wenn −IORQ auf Low liegt.

Wenn also auch noch A0 auf Low liegt und A1 auf Low, so wird Pin 4, IC6 auf Low gehen und IC2 schaltet durch. Wenn A0 auf High liegt und A1 auf Low, so schaltet IC3 durch. Die anderen beiden Kombinationen führen zu keiner Selektion. Bei einem Schreibvorgang werden IC2 und IC3 nicht selektiert, jedoch IC4 oder IC5. Diese beiden Bausteine sind sogenannte Latches. Sie können Daten zwischenspeichern. Mit der steigenden Flanke an Pin 11 werden die Daten übernommen. Die Bausteine verfügen auch über TRI-State-Ausgänge, die man über Pin 1 freischalten kann. Wenn an Pin 1 ein Low-Signal liegt, so sind die Ausgänge aktiv, sonst offen.

Im Prinzip kann man IC2 und IC3 auch zur Ausgabe verwenden, jedoch muß man dazu die -CS-Logik abändern. Das DIR-Signal ist schon richtig belegt. Allerdings können die Bausteine keine Werte zwischenspeichern, man verwendet sie z. B. zur Trennung von anderen Peripheriebausteinen, die man auf dem Verdrahtungsfeld aufbauen kann, doch das ist nur was für Fortgeschrittene.

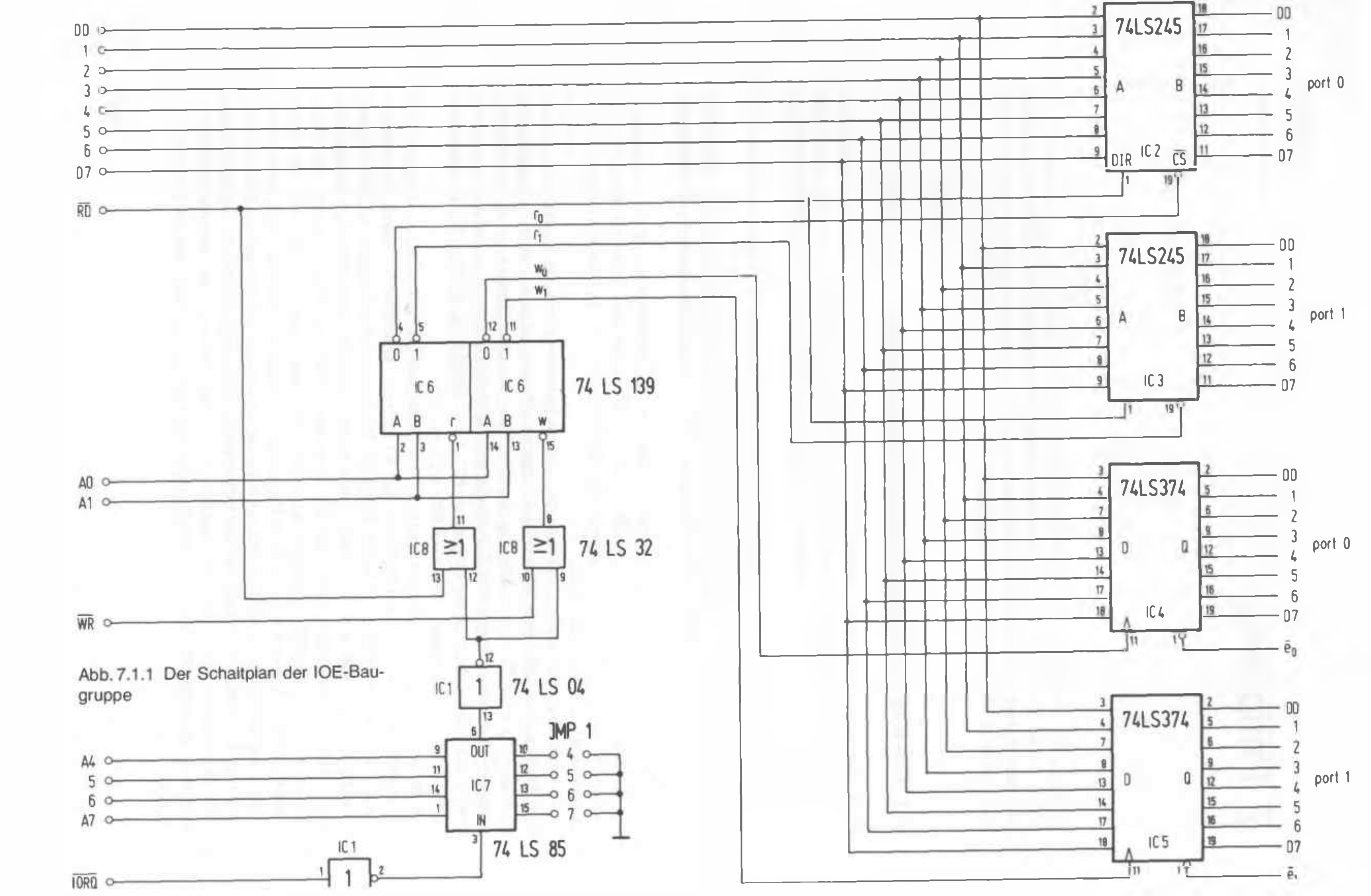


Abb. 7.1.1 Der Schaltplan der IOE-Baugruppe

M	o	o	M	
7	o	o	6	Port 0 (Eingang)
5	o	o	4	
3	o	o	2	
1	o	o	0	
7	o	o	6	Port 1 (Eingang)
5	o	o	4	
3	o	o	2	
1	o	o	0	
M	o	o	M	
6	o	o	4	Port 1 (Ausgang)
2	o	o	0	
1	o	o	3	
5	o	o	7	
6	o	o	4	Port 0 (Ausgang)
2	o	o	0	
1	o	o	3	
5	o	o	7	
$\overline{e1}$	o	o	$\overline{e0}$	
M	o	o	M	

	00	10	20	30	40	50	60	70
o 4 o	—	· ·	—	· ·	—	· ·	—	· ·
o 5 o	—	—	· ·	· ·	—	—	· ·	· ·
o 6 o	—	—	—	—	· ·	· ·	· ·	· ·
o 7 o	—	—	—	—	—	—	—	—

	80	90	A0	B0	C0	D0	E0	F0
o 4 o	—	· ·	—	· ·	—	· ·	—	· ·
o 5 o	—	—	· ·	· ·	—	—	· ·	· ·
o 6 o	—	—	—	—	· ·	· ·	· ·	· ·
o 7 o	· ·	· ·	· ·	· ·	· ·	· ·	· ·	· ·

Abb. 7.1.3 Die Einstellung von IO-Adressen

Abb. 7.1.2 (links) Die Anschluß-Belegung am Lochrasterfeld

Abb. 7.1.3 zeigt eine Zusammenstellung für die Einstellung der IO-Adressen an JMP1. Die Karte belegt immer 16 IO-Adressen, da A0, A1 an den Dekoder gehen und A2 und A3 nicht verwendet sind.

A4 bis A7 bestimmen die Adresse. Wenn man für die IO-Baugruppe zum Beispiel den Bereich 60 bis 6F haben möchte, so wird je eine Brücke bei 4 und bei 7 eingesetzt. Port 0 liegt dann auf Adresse 60h (h steht für hex, bzw. sedezimal), Port 1 liegt auf 61h. 62h und 63h sind nicht belegt, dann kommt aber wieder Port 0 auf 64h usw. Die Mehrfachbelegung kommt daher, daß A2 und A3 nicht in die Adressierung mit einbezogen sind. Wer das ändern will, muß ein paar Gatter mehr verwenden und sie entsprechend hinter IC1, Pin 12 schalten und das Signal mit A2 und A3 verknüpfen.

Abb. 7.1.4 zeigt die Lötseite der Leiterplatte, *Abb. 7.1.5* die Bestückungsseite und *Abb. 7.1.6* zeigt den Bestückungsplan. Eine Stückliste ist in *Tabelle 7.1.1* abgebildet.

Aufbau und Test der Baugruppe:

Zum Test benötigen Sie den Prüfstift und einen NDR-Klein-Computer mit Grundprogramm. Löten Sie zunächst alle Sockel ein. Dann bestücken Sie die Karte mit den ICs. Als Adresse für die Baugruppe wählen Sie 30h, also Brücken bei 6 und 7 einlöten. Ferner müssen Sie Brücken von -e0 und -e1 nach 0V legen, damit die Ausgänge der Latches aktiv sind.

Setzen Sie die Baugruppe ein und schalten den Computer ein.

Das Grundprogramm muß sich melden. Sollte das nicht sein, ausschalten und nach einem Kurzschluß auf der IOE suchen bzw. nach falsch eingesetzten ICs. Wenn es sich meldet, wählen Sie den Menüpunkt „IO setzen". Als Adresse geben Sie 30 ein. Als Datenwert geben Sie 0 ein. Wenn Sie nun mit dem Prüfstift die Ausgänge von Port 0 nachmessen, müssen alle Ausgänge auf Low sein.

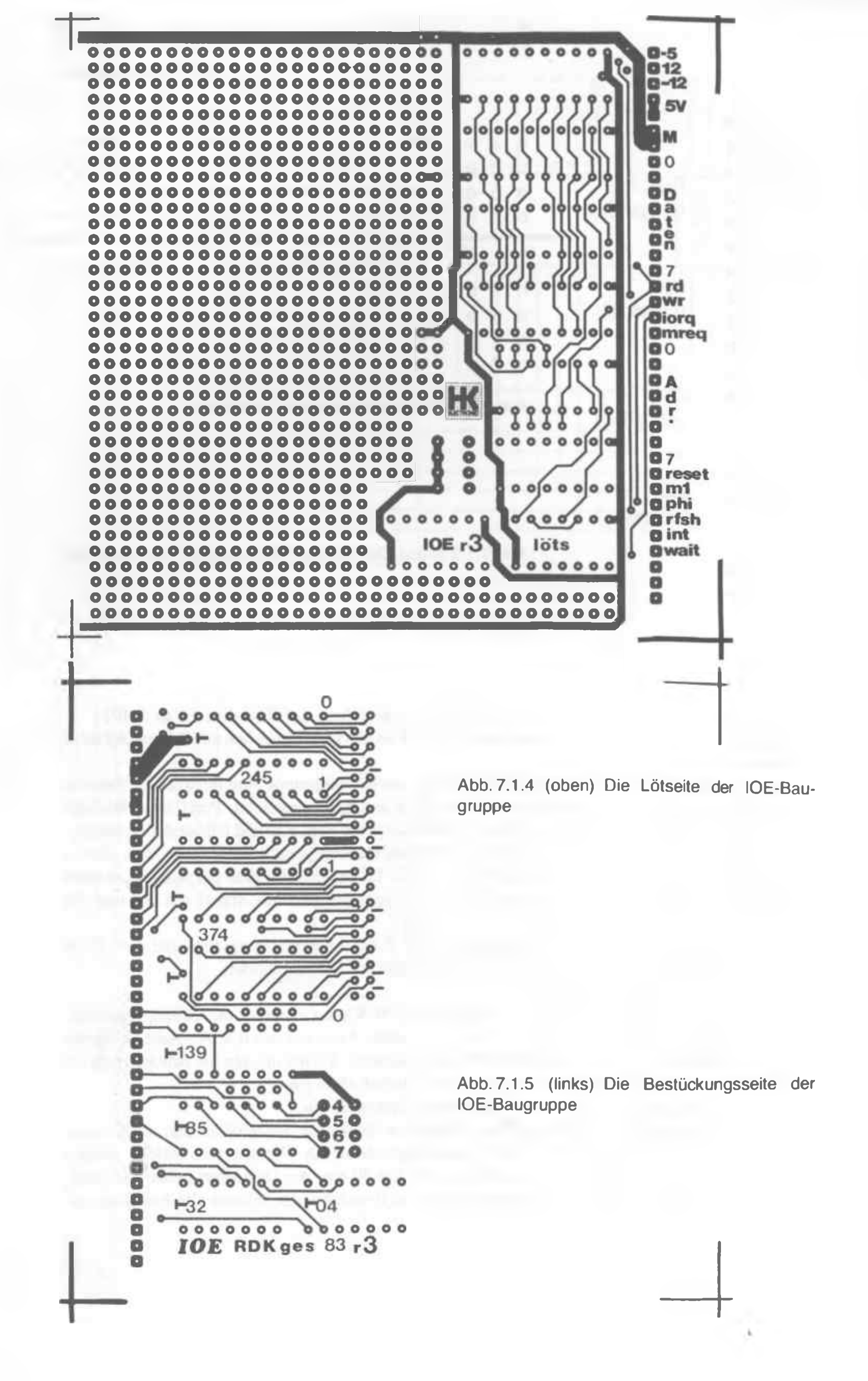


Abb. 7.1.4 (oben) Die Lötseite der IOE-Baugruppe

Abb. 7.1.5 (links) Die Bestückungsseite der IOE-Baugruppe

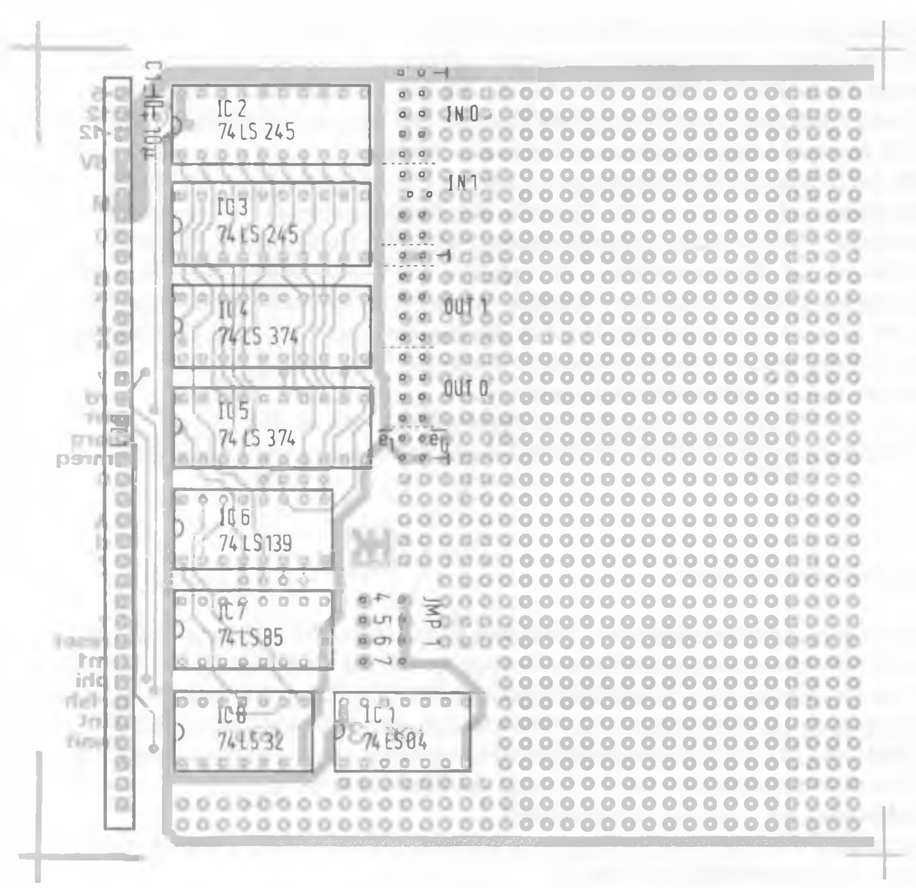


Abb. 7.1.6 Der Bestückungsplan der IOE-Baugruppe

Tabelle 7.1.1 Stückliste zur IOE-Baugruppe

IC 1 74LS04 Inverter
IC 2, IC 3 74LS245 bidirektionaler Bustreiber
IC 4, IC 5 74LS374 Zwischenspeicher mit TRI-State-Ausgang
IC 6 74LS139 zwei 1-aus-4 Dekoder
IC 7 74LS85 Vergleicher
IC 8 74LS32 Oder-Glied
4× 20polige IC-Fassung
2× 16polige IC-Fassung
2× 14polige IC-Fassung
C1 10 µF
1× 36polige Stiftleiste, gewinkelt, einreihig.
1× ges-IOE-Leiterplatte.
ggf.:
1× 50polige doppelreihige Stiftleiste, gerade.
Kenndaten:
Spannungsversorgung: + 5 V, 190 mA

Als nächstes gehen Sie wieder in das Menü „IO setzen“ und geben 30 ein. Nun geben Sie als Datenwert aber FF ein. Alle Ausgänge von Port 0 müssen nun einen High-Pegel haben.

So können Sie auch Port 1 testen, indem Sie die Adresse 31h eingeben. Achtung, obwohl die Werte alle sedezimal sind, geben Sie nur die Ziffern ein und kein nachfolgendes h. Das „h“ dient hier nur zur besseren Unterscheidung von dezimalen Werten.

Der Test ist natürlich nicht vollständig, Sie können jetzt aber noch jedes Bit einzeln testen, indem Sie nacheinander 1, 2, 4, 8, 10h, 20h, 40h, 80h eingeben (ohne das „h“ einzutippen). Nun müssen noch die beiden Eingänge getestet werden. Dazu rufen Sie das Menü „IO lesen“ auf. Als Adresse wird wieder 30h eingegeben. Das Ergebnis ist FF oder binär 11111111, denn alle Eingänge liegen auf High. Ändern Sie das, indem Sie an Port 0, Bit 0 ein 0-Signal legen, also den Eingang mit Masse (0V) verbinden. Drücken Sie die Taste „D“, um eine Dauereingabe zu bewirken. Diesmal erscheint 11111110 auf der Anzeige.

Dann testen Sie so nacheinander alle Eingänge.

Prüfen Sie auf die gleiche Weise auch den Port 1 mit der Adresse 31h.

Die IO-Karte ist fertig, doch was damit tun? Zunächst einmal eine ganz praktische Anwendung.

Die Ansteuerung eines Druckers:

Viele Drucker verwenden die sogenannte Centronics-Schnittstelle. Das sind einige Leitungen, die nach einem bestimmten Schema mit dem Drucker verbunden sind. *Tabelle 7.1.2* zeigt das Material, das Sie neben einem Drucker dazu benötigen. *Abb. 7.1.7* zeigt das Verdrahtungsschema der IOE-Baugruppe. Die IOE-Karte wird auf die Adressen 40h..4Fh gelegt, die im BASIC, GOSI sowie ASSEM und FLOMON eingebauten Druckerprogramme verwenden die Adresse 48h, 49h für die Centronics-Schnittstelle. *Abb. 7.1.8* zeigt die Brückeneinstellung.

Wie funktioniert die Centronics-Schnittstelle?

Zeichen werden im ASCII-Format (siehe KEY-Beschreibung) an den Drucker übertragen. Dazu dienen die Datenleitungen D0 bis D7.

Mit der Leitung -STROBE wird dem Drucker gesagt, wann die Daten gültig sind. Da der Drucker zum Drucken auch einige Zeit braucht, besitzt er auch eine Leitung, mit der er dem Computer sagen kann, ob er für Daten bereit ist. Sie trägt den Namen BUSY.

Wenn man Daten an den Drucker schicken will, muß man also zuerst die Leitung BUSY abfragen. Ist sie auf High, so ist der Drucker beschäftigt und man muß warten. Ist sie aber auf Low, so darf der Daten schicken. Der Computer legt die Daten an, und gibt einen kurzen Low-Puls auf die Leitung -STROBE.

Abb. 7.1.9 zeigt einen solchen Vorgang. BUSY liegt auf Low, also können zuerst die Daten übertragen werden. Das geschieht durch einen kurzen Puls an IC5, Pin 11, also Port 0. Danach wird der STROBE-Puls ausgegeben. Man sieht, daß -STROBE auf 0 geht. Gleich darauf geht BUSY auf High, denn der Drucker hat nun ein Zeichen bekommen und ist damit beschäftigt. Das STROBE-Signal geht wieder zurück auf High.

Tabelle 7.1.2 Stückliste zur Centronics-Drucker-Schnittstelle

1 × IOE-Baugruppe
1 × Centronics-Buchse oder Stecker 36polig (Fachhandel fragen). Eine Buchse, wenn man ein konfektioniertes Verlängerungskabel mit zwei Steckern zusätzlich verwendet, einen Stecker, wenn man diesen direkt verdrahten will.
1 × ggf. Verlängerungskabel.

Abb. 7.1.7 Verbindungsschema

Abb. 7.1.8 (rechts) Die Brückeneinstellung

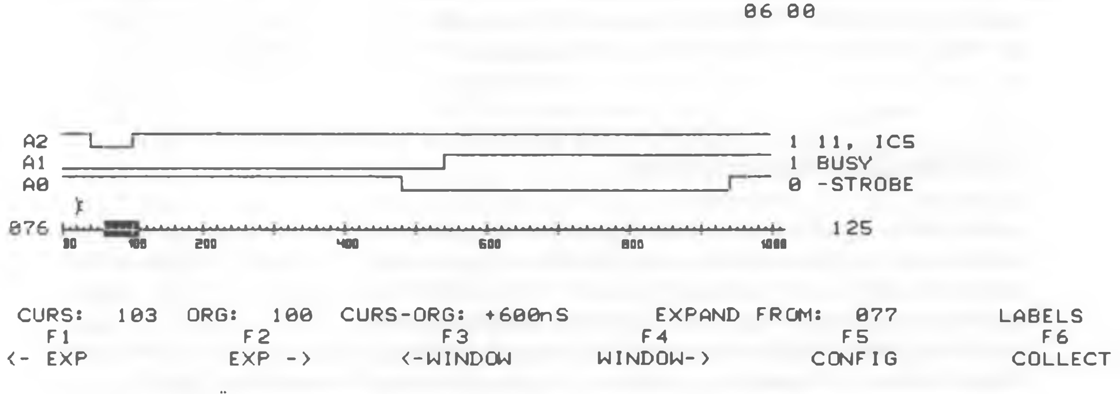


Abb. 7.1.9 Die Übertragung eines Bytes

10uS CLOCK MAG: 5X SCRN INTV: 2.00mS MAIN MEMORY
05 00



Abb. 7.1.10 Die Übertragung von Daten

Abb. 7.1.10 zeigt die Übertragung mehrerer Bytes. Der Drucker besitzt selbst einen kleinen Datenpuffer, so daß die Übertragung zunächst einmal schneller geht als der Drucker zum Drucken braucht. Dann, hier nicht sichtbar, werden auch längere BUSY-Pausen eingelegt.

Ein Beispiel-Programm zur direkten Ansteuerung könnte wie folgt aussehen:

```
DB 49   Warte:   IN A, (49H)      einlesen des BUSY-Status
E6 01            AND 1            Bit 0, isolieren
20 FA            JR NZ, Warte     Warten bis BUSY auf Low
3E 42            LD A, ‚B‘        Buchstaben „B“ in Akku laden
D3 48            OUT (48H), A     Buchstabe auf Datenport geben
3E 01            LD A, 1          STROBE ---___---- ausgeben
D3 49            OUT (49H), A     zuerst High, sicherheitshalber
3E 00            LD A, 0          dann Low
D3 49            OUT (49H), A     ausgeben
3E 01            LD A, 1          dann wieder High
D3 49            OUT (49H), A     ausgeben
18 E8            JR Warte         Wiederholung durchführen
```

Geben Sie dieses Programm ab Adresse 8800 ein. Wer die Z80 Vollausbau-CPU besitzt, kann auch den Zeilenassembler verwenden. Ein angeschlossener Drucker wird lauter B ausdrucken. Stoppen Sie das Programm dann durch Reset. Auf diese Weise kann man den Drucker betreiben.

Wenn Sie Texte ausgeben, so benötigt man neben den Zeichencodes für A..Z usw. auch noch Steuerzeichen. Mit dem Code 0Dh kann man den Druckkopf an den Zeilenanfang zurückfahren lassen, mit dem Code 0Ah macht er einen Zeilenvorschub. Weitere Codes, die ansonsten vom jeweiligen Druckertyp abhängen, sind in Ihrem Druckerhandbuch beschrieben.

7.2 Die CAS-Baugruppe

Bisher waren Programme im Computerspeicher immer verloren, wenn die Spannung des NDR-Klein-Computers ausgeschaltet wurde. Wertvolle Programme muß man aber abspeichern können. Dazu soll ein Kassettenrecorder dienen. Hier werden die Elektronik geschildert und die Software, mit der man Programme und Daten aufzeichnen kann.

Daß man auf Tonbändern Musik aufzeichnen kann, weiß jeder, daß man aber auch Daten für Computer darauf ablegen kann, ist nicht so bekannt. Allerdings kann man Daten nicht so ohne weiteres auf das Tonband bringen, sie müssen dazu aufbereitet werden. Dafür benötigt man eine

spezielle Schaltung, ein Interface, die Kassetten-Baugruppe. Sie arbeitet folgendermaßen: Zunächst werden die Daten-Bytes, die im Computerspeicher liegen, durch das Grundprogramm nacheinander in einen Baustein (mit der Bezeichnung 6850) auf die CAS-Baugruppe befördert. Dieser Baustein zerlegt die angelieferten Bytes in die einzelnen Bits und überträgt diese einzeln hintereinander an die restliche Schaltung. Die Bits tragen die Information 0 oder 1, in Spannungswerten ausgedrückt: 0 V oder 5 V.

Man kann nun einmal versuchen, diesen Datenstrom direkt an den Tonbandeingang eines Kassettenrecorders zu führen. Dann wird man feststellen, daß bei einer Wiedergabe alles andere als das ursprüngliche Signal wieder herauskommt. So einfach geht es also nicht. Eine spezielle Schaltung muß das Datensignal so umwandeln, daß es ein Tonsignal wird. Erst dann wird es dem Kassettenrecorder zugeführt. Eine solche Schaltung nennt man Modulator. Umgekehrt muß aus einem solchen Signal auf Tonband das ursprüngliche Bitmuster wieder zurückgewonnen werden. Man sagt „es wird demoduliert". Anschließend werden die Bits wieder in einen Strom von Bytes verwandelt, die in den Speicher des Computers zurückgeladen werden können.

Man kann Daten nicht beliebig schnell an einen Kassettenrecorder übertragen, da ein Kassettenrecorder Schwingungen, die eine bestimmte maximale Frequenz überschreiten, nicht mehr aufzeichnen kann. Die Zahl der Bits, die pro Sekunde übertragen werden, ist eine wichtige Größe. Sie wird als Baudrate bezeichnet. Die CAS-Baugruppe, die gleich aufgebaut werden wird, ist für eine Übertragungsrate von 1200 Baud ausgelegt. Das bedeutet: 1200 Bit pro Sekunde können übertragen werden. Eine höhere Baudrate ist zwar möglich (bis zu 6000 Baud), jedoch benötigt man dann ausgesuchte Kassettenrecorder. *Abb. 7.2.1* zeigt den Schaltplan der Kassettenschaltung, *Abb. 7.2.2* die Lötseite, *Abb. 7.2.3* die Bestückungs-Seite und *Abb. 7.2.4* den Bestükkungsplan. *Tabelle 7.2.1* zeigt die Stückliste.

So funktioniert CAS

Es sei ganz grob geschildert, wie die Schaltung arbeitet. Der Baustein 6850 wird vom Computer über eine I/O-Adresse angesprochen, die mit dem Adreßdecoder, bestehend aus den beiden 7485-Bausteinen, eingestellt werden kann. Wenn der Computer ein Byte abgeliefert hat, sorgt eine spezielle Schaltung im 6850 dafür, daß dessen einzelne Bits zunächst vorn um eine 0 und hinten um zweimal 1 ergänzt werden. Ein Taktgenerator, bestehend aus dem Baustein 555 liefert einen präzisen Takt, der noch auf 1200 Hz halbiert wird und dem 6850-Baustein zugeführt wird. Im Baustein wird während einer Periode des Taktsignales nun immer je ein Bit des angelieferten und ergänzten Bytes genommen, beginnend bei der zugesetzten Null, dem Startbit, und dieses am Ausgang Txd ausgegeben. Präzise und synchron mit dem Taktsignal erscheint also am Ausgang die Folge der einzelnen Datenbits, vornweg eine Null, hinten daran zwei Einsen, die einem späteren Empfänger Zeit lassen sollen, das Empfangene Byte auch wieder auszuwerten. Takt- und Datensignal sind beide an einen EXOR-Baustein geführt und werden dort gemischt. Dabei geschieht Entscheidendes (*Abb. 7.2.5*). Ein EXOR mit zwei Eingängen führt am Ausgang genau dann eine 1, wenn genau einer der Eingänge 1 ist. Dies führt dazu, daß bei einer Mischung von Daten- und Taktsignal am EXOR immer dann, wenn kein Signalwechsel bei dem Datensignal stattfindet und dieses auf 0 liegt, der Originaltakt anliegt. Wechselt zu Beginn einer Periode das Datensignal von 0 auf 1, so wird im gemischten Signal genau die Hälfte des Taktsignales zusätzlich gewartet, bis dort ein Polaritätswechsel von 0 auf 1 stattfindet. Dies geschieht analog, wenn das Datensignal von 1 auf 0 wechselt. Da der Baustein 6850 die Datenbits exakt synchron mit den Perioden der Taktschwingung abgibt, kann man, wenn man das Mischungsergebnis in der zweiten Hälfte einer jeden Takt-Periode anschaut, feststellen, ob von 0 auf 1 gewechselt wurde, denn dann ist dort der Wert 1 festzustellen, oder der Wert 0, wenn von 1 auf 0 gewechselt wurde.

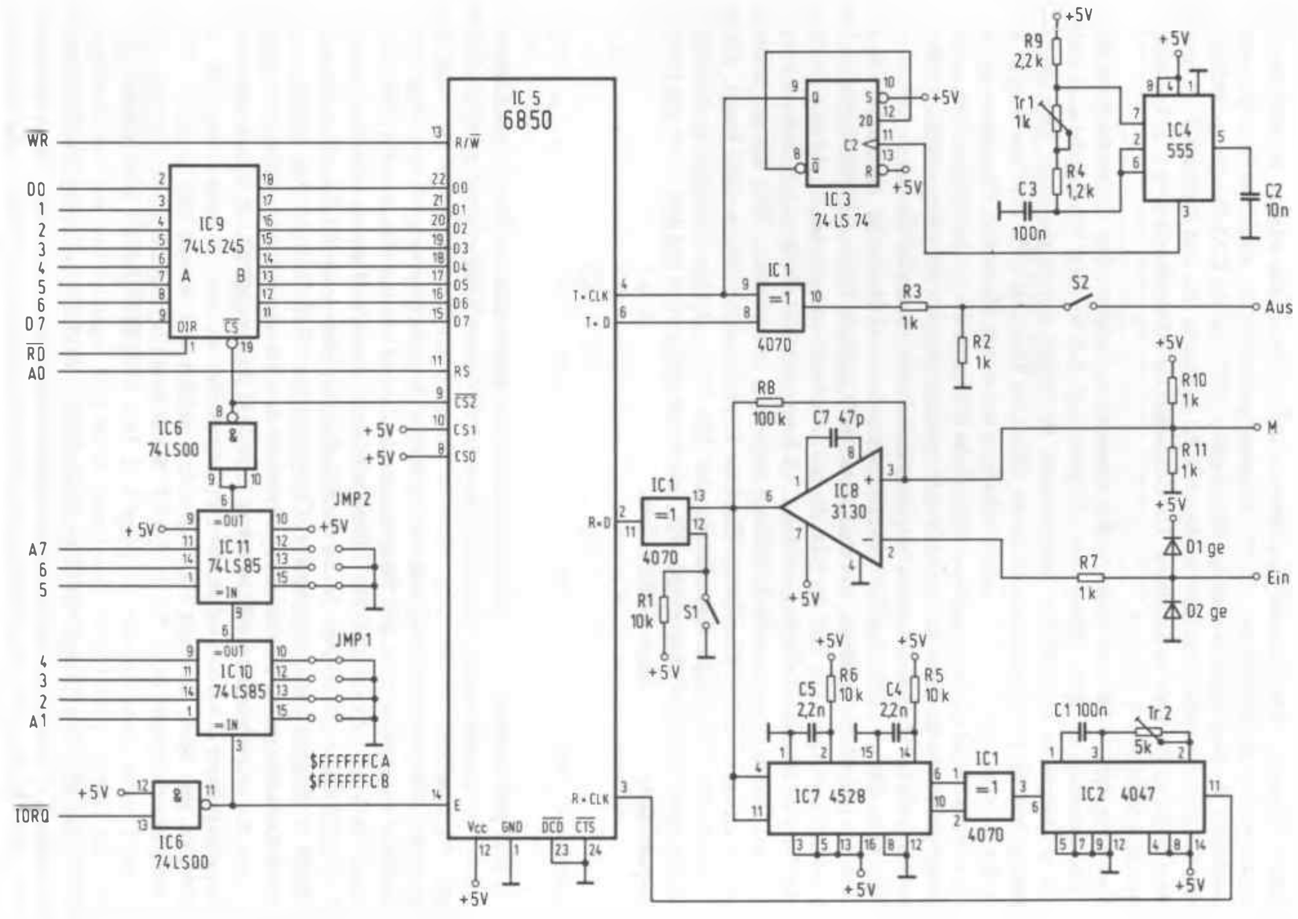


Abb. 7.2.1 Das Schaltbild der Baugruppe CAS

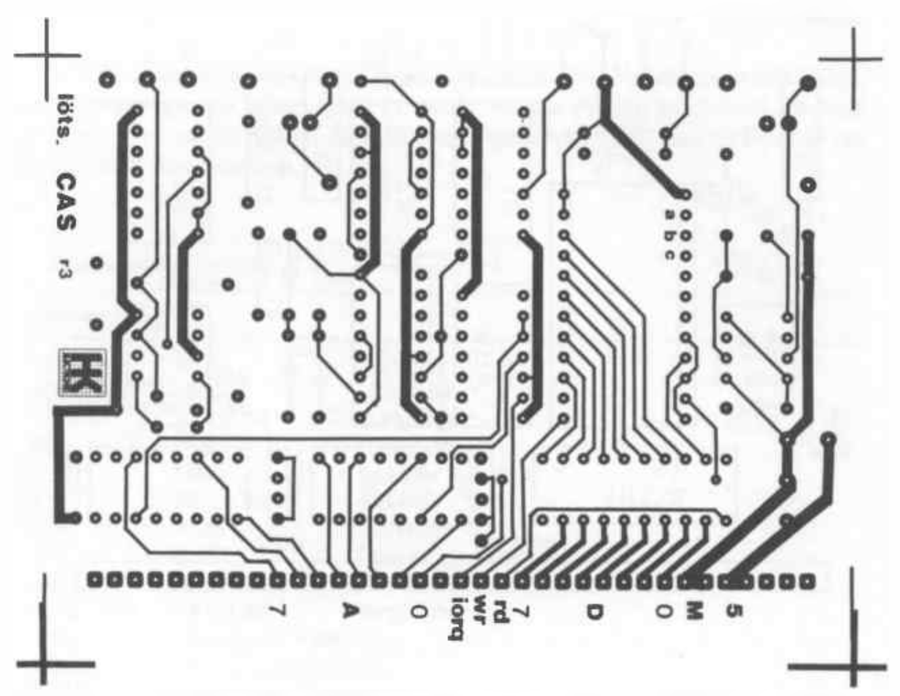


Abb. 7.2.2 Die Lötseite der Leiterplatte CAS

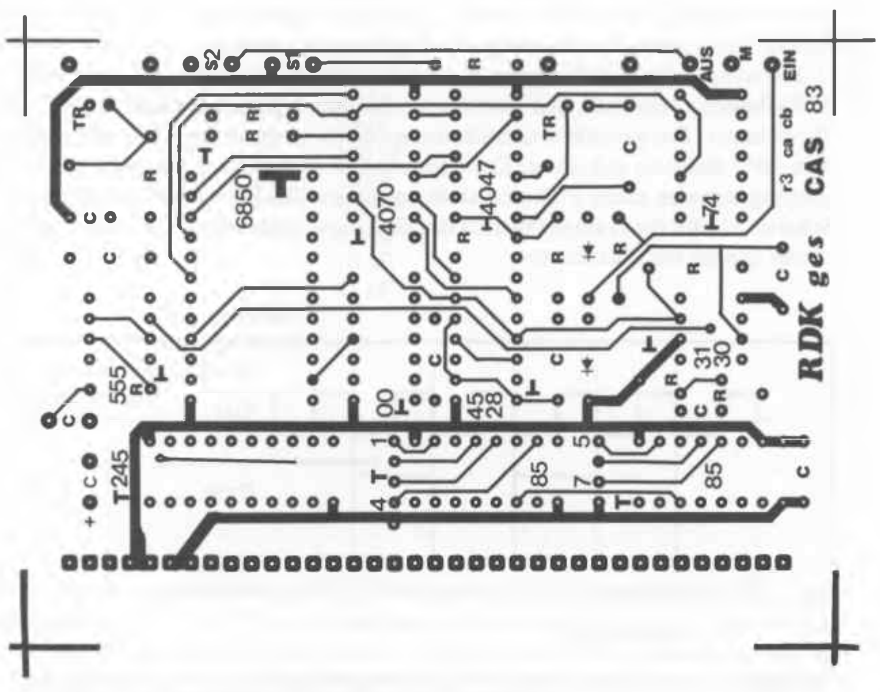


Abb. 7.2.3 Die Bestückungsseite der Leiterplatte CAS

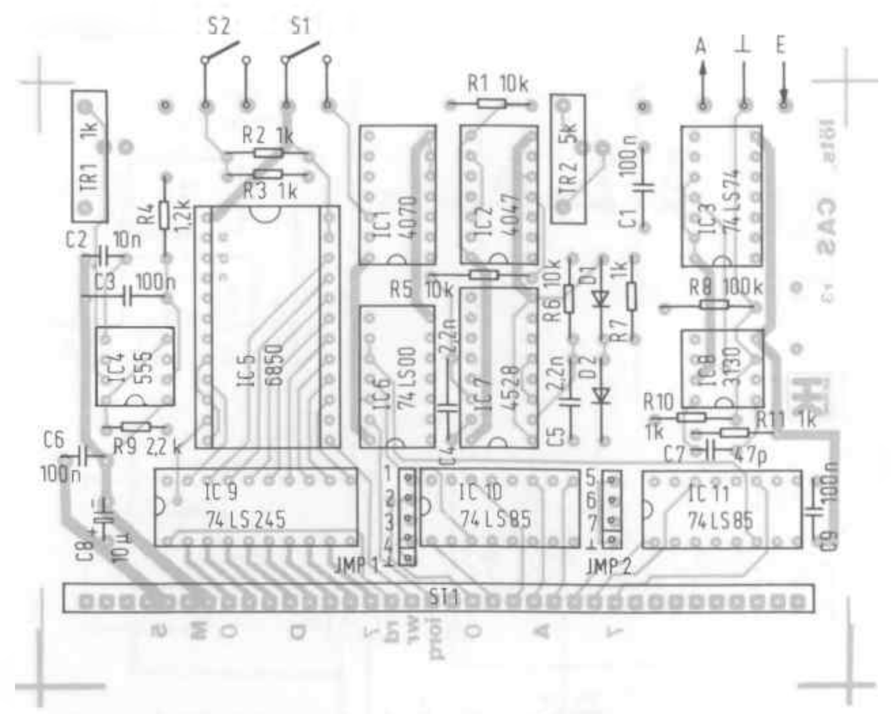


Abb. 7.2.4 Der Bestückungsplan der Baugruppe CAS

Genau so etwa funktioniert die Auswerteschaltung, bei der zunächst das Signal verstärkt wird und dann im unteren Teil an Pin 11 des 4047 ein Signal erzeugt wird, das exakt die Mitte einer jeden Periode der Sendetaktschwingung aus dem modulierten Signal entnimmt. Das Monoflop MV3 (J2) ist so eingestellt, daß es genau ¾ einer Periode abwartet und dann einen Puls erzeugt, der den 6850-Baustein veranlaßt, den Wert des modulierten Signales an dieser Stelle am Eingang Rxd zu übernehmen. Das geschieht nämlich immer dann, wenn an Rx CLK eine steigende Flanke aus dem 4047-Baustein ankommt. Der Baustein CD 4528 erzeugt übrigens bei jedem Wechsel des Eingangssignales einen Puls und stellt so immer den Bezug zur Sendetakt-Periode fest. Der Schalter S1 hilft die richtige Polarität des Eingangssignales einstellen, denn manche Kassettenrecorder kehren die Polarität um.

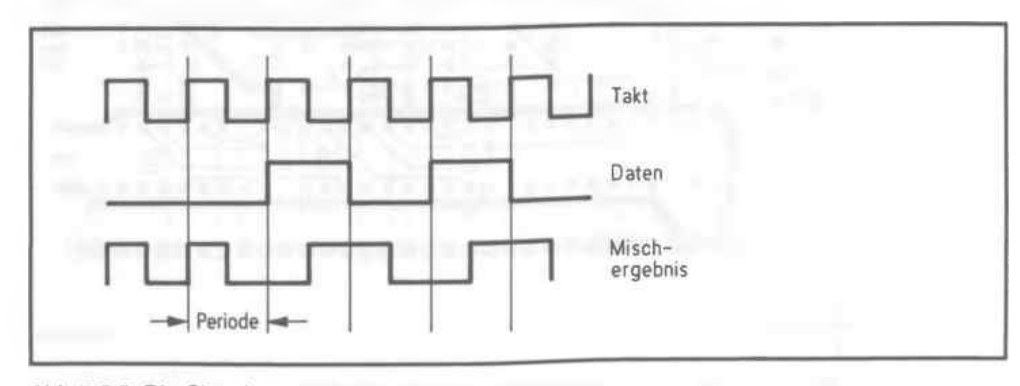


Abb. 7.2.5 Die Signalerzeugung

Wenn Sie mit diesen Angaben versuchen, einen Impulsplan des Verfahrens aufzustellen, werden Sie noch Schwierigkeiten haben, denn es wurde nur das Prinzip geschildert. Es muß nämlich bei der Auswertung auf die richtige Signalflanke eingerastet werden, und die Polarität des Wiedergabesignales muß invertiert sein.

Tabelle 7.2.1 Die Stückliste zu CAS

Stück	Bezeichnung		
1	IC1	4070	4 EXOR
1	IC2	4047	Monoflop
1	IC3	74 LS 74	D-Flipflop
1	IC4	555	Generator
1	IC5	6850	Serieller Baustein
1	IC6	74 LS 00	4 NAND
1	IC7	4528	Monoflop
1	IC8	3130	Operationsverstärker
1	IC9	74 LS 245	Bus-Transceiver
2	IC10, IC11	74 LS 85	Vergleicher
2	SO8	8polige IC-Fassung	
4	SO14	14polige IC-Fassung	
3	SO16	16polige IC-Fassung	
1	SO20	20polige IC-Fassung	
1	SO24	24polige IC-Fassung	
3	R1, R5, R6		10 kΩ
5	R2, R3, R7, R10, R11		1 kΩ
1	R8		100 kΩ
1	R4		1,2 kΩ
1	R9		2,2 kΩ
1	Tr1		1 kΩ
1	Tr2		5 kΩ
2	C1, 3		100 nF
1	C2		10 nF
2	C9, 6		100 nF
2	C4, 5		2,2 nF
1	C8		10 µF
1	C7		47 pF
2	D1, 2	Siliziumdioden	
1	St1	36polige Steckerleiste	
1	Platine mit Lötstopplack		
1	Dioden-Buchse		

Wie man CAS justiert

In der Schaltung befinden sich zwei Schalter. S2 wird immer dann eingeschaltet, wenn eine Aufnahme durchgeführt werden soll. S2 leitet das Signal an den Kassettenrecorder. Bei manchen Kassettenrecordern wird das analoge Signal zwischen Aufnahme und Wiedergabe invertiert, bei anderen nicht. Um das Signal immer in die richtige Polarität zu bringen, gibt es den Schalter S1. Die richtige Lage muß man ausprobieren. Doch nun zum Abgleich.

```
RDK-Digital-Scop

1 = Periodendauer,      1-Kanal
2 = Vergleichsmessung, 2-Kanal
3 = Kurzerklaerung
```

Abb. 7.2.6 Das Menü zum Skop-EPROM

```
*** Rev 1.0 ***
1. IOE-Karte  auf Adresse 30h legen.
2. I/O 0-Port Bit 0 = Kanal 1.
3. I/O 0-Port Bit 1 = Kanal 2.
4. Kanal 1 ist auch Triggereingang.
5. Das Signal erscheint erst
   nachdem ein Signalwechsel
   am Triggereingang von 0 auf 1
   erfolgt.
6. Die Messungen erfolgen auf ca.
   5ys bis 6ys genau.
                    M=Menue
```

Abb. 7.2.7 Die Kurzerklärung der Funktionen im EPROM

Dazu wird ein Programm verwendet, das es ermöglicht, aus dem SBC2-Computer ein Oszilloskop zu machen. Das SKOP-EPROM wird dazu in Fassung 3 (IC9) auf der SBC2-Baugruppe anstelle des RAM-Bausteins eingesteckt. Dann wird der Computer eingeschaltet, es meldet sich wie bisher das Grundprogramm. Das Skop-Programm im EPROM 2716 muß jetzt gestartet werden. Dazu gehe man ins Startmenü und gebe die Adresse 8800 an. Dann meldet sich ein SKOP-Menü *(Abb. 7.2.6)* Dort gibt es drei Menüpunkte. Der dritte liefert eine aktuelle Kurzerklärung, die wie in *Abb. 7.2.7* aussieht.

Man benötigt noch eine IOE-Karte, um die Messungen durchführen zu können. Diese wird auf die Adresse 30 eingestellt (also Brücken 7 und 6 einlöten). Es wird das IC 74LS245, das ganz am Rand sitzt, benutzt. Die Bits 0 und 1 sind die Meßeingänge (siehe IOE-Schaltplan). *Abb. 7.2.8* zeigt die Belegung an dem Benutzerstecker der IOE-Karte. Wenn man dort Stifte eingelötet hat, sollte man zwei Meßleitungen auf der Lötseite der IOE-Karte anbringen oder Buchsen verwenden.

Der Computer hilft beim Abgleich

Nun zum Abgleich. Es muß zuerst der Sendetakt abgeglichen werden. Dazu gibt es das Trimm-Potentiometer TR1 auf der CAS-Baugruppe. Man muß die Meßleitung Kanal 1 mit dem Sendetakt verbinden. Der Sendetakt findet sich auf der Lötseite der CAS-Baugruppe an Pin 4 des ICs 6850. Dort ist der kleine Buchstabe „c“ aufgedruckt. Man lötet oder klemmt die Meßleitung dort fest. Nun schaltet man den Computer ein und startet bei Adresse 8800. Dann wählt man Menü 1 „Periodendauer 1-Kanal“ aus. Jetzt muß auf dem Bildschirm ein Rechtecksignal sichtbar sein. Wenn nicht, so muß man alle Verbindungen überprüfen und auch die Brücken auf der IOE-Karte.

Man darf auch Kanal 1 nicht mit Kanal 2 verwechseln. Mit einem Schraubendreher wird dann TR1 solange verdreht, bis sich *Abb. 7.2.9* einstellt. Die gemessene Periodendauer muß etwa 833 Mikrosekunden betragen. Aufgrund der Meßunsicherheit wird man aber nur eine Näherung einstellen können, was aber ausreicht.

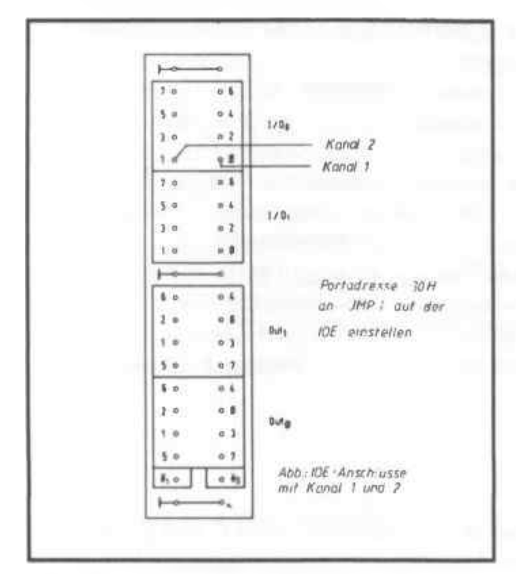


Abb. 7.2.8 Mit diesen Anschlüssen auf der IOE-Platine muß CAS verbunden werden. Die IOE-Platine muß auf 30 adressierbar sein

Periode = 0835 ys

S=Speichern W=Weiter M=Menue

Abb. 7.2.9 So sieht ein Rechtecksignal aus, mit dem Skop-EPROM auf dem Bildschirm dargestellt, nachdem TR1 abgeglichen wurde

Jetzt muß der Empfangsteil abgeglichen werden. Dazu muß man S2 auf Aufnahme stellen. Man benötigt ferner eine kleine Hilfsschaltung, um den Abgleich zu vereinfachen. *Abb. 7.2.10* zeigt die Schaltung. Sie wird mit der CAS-Baugruppe verbunden.

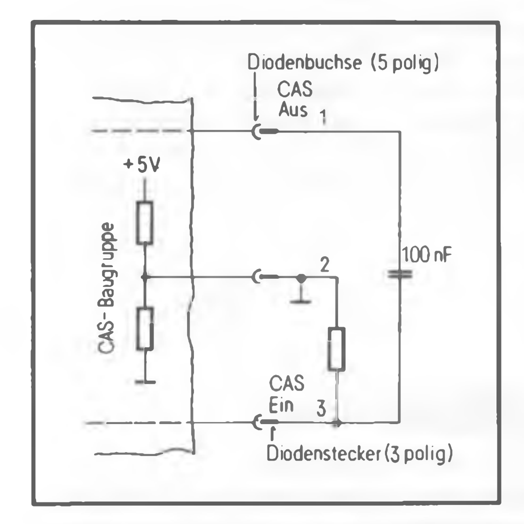


Abb. 7.2.10 So sieht die Selbst-Test-Schaltung aus. Die Steckverbinder sollten auf der CAS-Seite eine DIN-Buchse (Diodenstecker) und auf der Schaltungsseite ein entsprechender dreipoliger Stecker sein

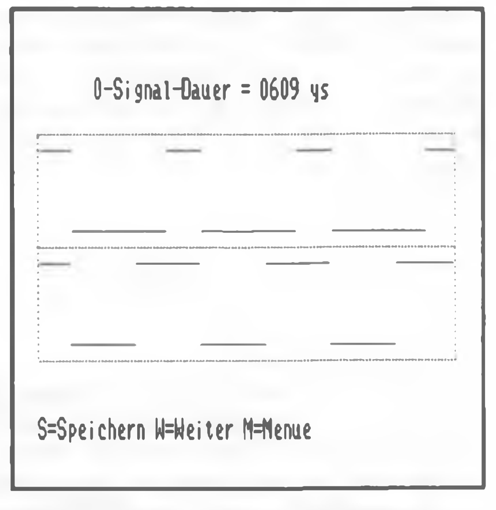


Abb. 7.2.11 So etwa muß der Test auf dem Bildschirm aussehen

Nun wird der Kanal 1 mit dem Meßpunkt b, das ist Pin 3 des 6850 und Kanal 2 mit dem Meßpunkt a, das ist Pin 2 des 6850 (CAS-Baugruppe), verbunden. Man wählt den Menüpunkt 2, „Vergleichsmessung 2-Kanal", an und kann den Abgleich an TR2 durchführen. Dabei wird die 0-Signal-Dauer gemessen. Sie muß auf etwa 625 Mikrosekunden eingestellt werden. Der genaue Wert ist nicht so kritisch, die Einstellung mit dem besten Näherungsergebnis ist genau die richtige. *Abb. 7.2.11* zeigt, wie es aussehen kann. Hier kann man auch die Funktion des Schalters S1 testen. Wenn man ihn umschaltet, so ändert sich die Polarität des unteren Signals.

Mit dem SKOP-EPROM kann man noch andere einfache Messungen durchführen, dabei ist die Frequenz auf etwa 190 kHz begrenzt, die minimale Frequenz auf etwa 370 Hz. Mit der Taste „S" kann man die Pulsform eines Signales speichern und in Ruhe betrachten. Die Triggerung, das ist die Auslösung des Meßvorgangs, wird automatisch bei einer Signal-Änderung am Kanal ausgeführt. Ein Meßergebnis erscheint nur, wenn die Frequenz im zulässigen Bereich liegt.

Der Kassettenrecorder wird angeschlossen

Man verbindet CAS-AUS mit dem Eingang des Kassettenrecorders und CAS-EIN mit dem Ausgang des Recorders und CAS-Masse mit der Masseleitung des Kassettenrecorders. ACHTUNG! Die Leitung CAS-Masse hat einen Spannungspegel von etwa 2.5 V, es ist dies nicht die Masse des Computers (0V). Man darf sie nicht verwechseln, sonst funktioniert die Wiedergabe nicht.

Mit dem SKOP-EPROM kann man auch die Wiedergabe durch das Tonbandgerät kontrollieren. Man muß dazu eine Datenaufzeichnung durchführen.

Abb. 7.2.12 zeigt die Belegung einiger gebräuchlicher Buchsen, entnommen den „Franzis Mini-Tabellen". Hier muß man etwas experimentieren, bis man seinen Kassettenrecorder richtig mit unserem Computer verbunden hat.

Beim Recorder sollte man darauf achten, daß er ein Bandlaufzählwerk und eine manuelle Aussteuerung besitzt. Manche Recorder arbeiten leider nicht so gut, da sie über Sprachfilter oder Entzerrer verfügen, die das CAS-Signal verfälschen. Es kommt nämlich auf dessen Phasenlage an. *Abb. 7.2.13* zeigt eine Schaltung, die man verwenden kann, wenn es große Probleme gibt. Sie rundet das CAS-Signal ab, was von manchen Recordern besser verdaut werden kann. Ein Recorder, der alle unsere Forderungen erfüllt, ist der Typ MK 2000 von der Firma Waltham, München. Der Preis beträgt etwa 85 DM. Weitere Typen können von den Herstellern der Bausätze erfragt werden.

Die erste Datenaufzeichnung

1. Computer einschalten.
2. Mit „W" wird das Menü mit den Kasettenfunktionen angewählt.
3. Dann „2" für „2 = Speichern CAS" eingeben.
4. Es meldet sich das Kassettenmenü. *Abb 7.2.14* zeigt den Bildschirm.
5. Als erstes wird nach einer Startadresse gefragt. Zum Test die Adresse 8800 einzugeben.
6. Als Endadresse wird 88FF eingegeben, das soll die letzte Adresse sein, bis zu der abgespeichert werden soll.
7. Dann wird nach einem Namen gefragt. Jetzt muß man dem abzuspeichernden Inhalt einen Namen geben, der später, zur Kontrolle, wieder angezeigt werden wird. *Abb. 7.2.15* zeigt den jetzigen Bildschirminhalt.

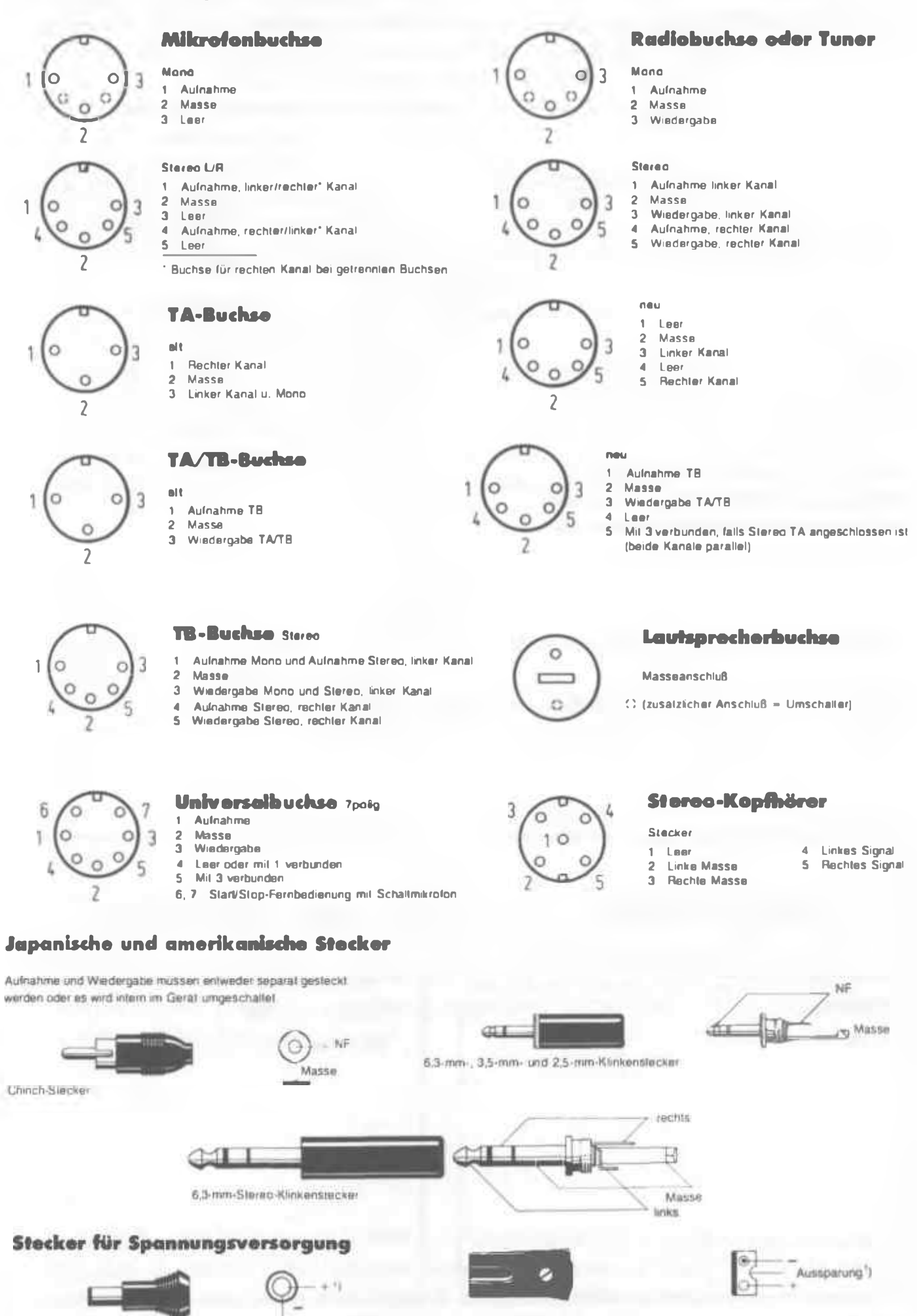


Abb. 7.2.12 Das sind einige Varianten von Steckverbindern und Anschlußbeschaltungen, wie sie in modernen Kassettenrekordern auftreten können

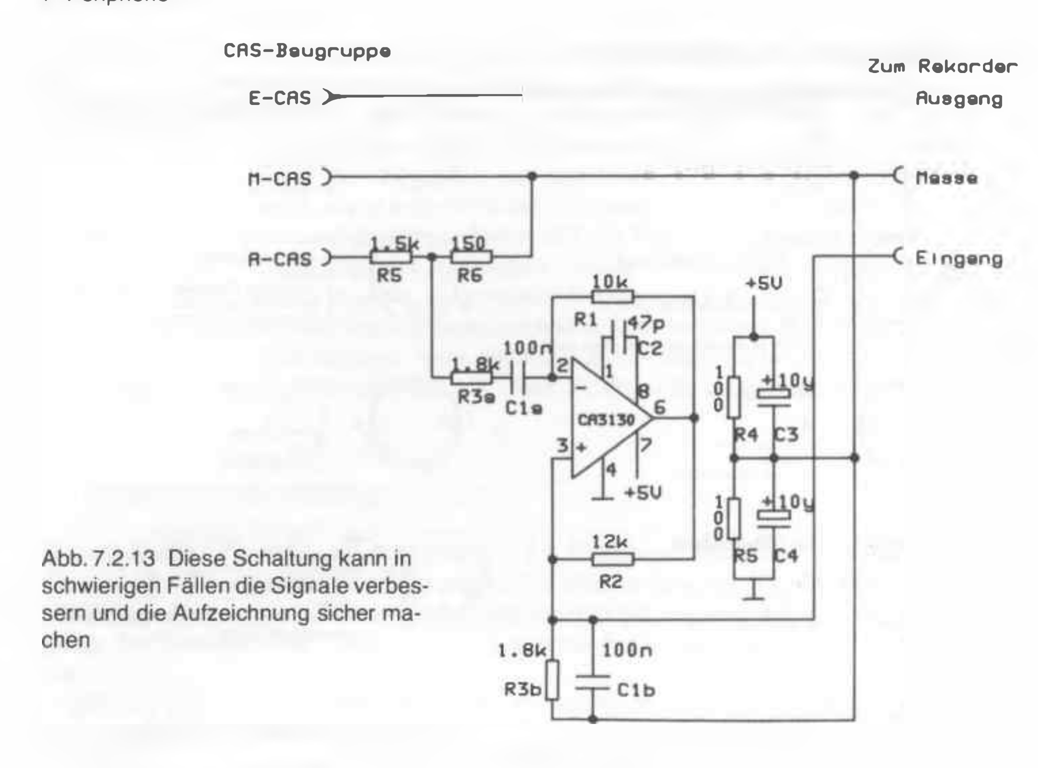


Abb. 7.2.13 Diese Schaltung kann in schwierigen Fällen die Signale verbessern und die Aufzeichnung sicher machen

auf CAS speichern

Startadr. :
Endadr. :
Name :

Abb. 7.2.14 Startadresse, Endadresse und Namen des Datenpaketes werden vom Menüpunkt "auf CAS speichern" abgefragt

auf CAS speichern

Startadr. : 8800
Endadr. : 88ff
Name : Test Aufzeichnung

Abb. 7.2.15 Gleich beginnt die Aufzeichnung

8. Ehe man nach der letzten Eingabe die Taste „CR" drückt, muß man den Kassettenrecorder auf Aufnahme schalten und starten. S2 muß geschlossen sein. Der Aussteuerungsanzeiger auf dem Kassettenrecorder sollte nun schon einen Pegel anzeigen.
9. Jetzt die Taste „CR" drücken, die Aufzeichnung beginnt.
10. Nach ein paar Sekunden zeigt sich wieder das Grundmenü.
11. Nun den Recorder zurückspulen.
12. Den Schalter S2 ausschalten.
13. Menüpunkt „3 = Prüfen CAS" anwählen.
14. Den Recorder starten.
15. Wenn keine Reaktion erfolgt, Schalter S1 umschalten und den Recorder zurückspulen. Neu starten.
16. *Abb. 7.2.16* zeigt, wie es aussehen soll, wenn die Daten erfolgreich geladen wurden.

Wenn keine Reaktion erfolgt, kann man nochmals mit dem Skop-EPROM messen. Diesmal aber ohne den Teststecker. Auf dem Bildschirm sieht man dann die Signalform. Wenn kein Signal vom Recorder erscheint, sollte man zunächst alle Leitungen sorgfältig kontrollieren. Manchmal sind auch die Aufnahmesignale und Wiedergabesignale eines Recorders auf die gleiche Leitung gelegt. Man muß dann CAS-EIN mit CAS-AUS verbinden.

Jetzt wird programmiert

Nun zur Programmierung, denn jetzt kann man auch größere Programme schreiben, ohne daß sie nach dem Abschalten verloren sind.

Wenn man nach dem bisherigen Verfahren unterschiedlich große Quadrate programmieren wollte, dann mußte man für jedes Quadrat ein eigenes Programmstück schreiben. Das soll sich ändern. Dazu benötigen wir aber neue Befehle.

Lade indirekt

Im Befehl 2A xxxx.W steht xxxx für eine Zahl oder einen Namen, wie beim Lade-Wert-Befehl. Jetzt gibt die nachfolgende Zahl aber nicht den Wert an, der geladen werden soll, sondern sie ist eine Speicheradresse. Bei Ausführung des Befehls holt sich die CPU (Z80) von zwei aufeinanderfolgenden Speicherzellen, beginnend bei der angegebenen Adresse die Zahl, die dann geladen wird. Ein Beispiel zeigt schon alles:

```
8800:     2A 8900.W
          CD SCHREITE

8900:     #123.W
```

Auf Adresse 8800 steht der Befehl „2A 8900.W". Wenn der Befehl ausgeführt wird, so wird der Inhalt der Speicherzellen 8900, 8901 geladen. Dort steht der dezimale Wert 123, wie die Anzeige ausweist. Also ist die Anweisung jetzt identisch mit dem Befehl „21 #123.W". Weshalb zwei Speicherzellen? Weil der Lade-indirekt-Befehl eine 16-Bit-Größe lädt, genauso, wie auch der Lade-Wert-Befehl. Wenn man einen Zahlenwert mit „W" abschließt, werden automatisch zwei Speicherzellen verwendet. Deshalb kann man auch einfach #123.W als Datenwert angeben.

```
Prueflesen CAS

Test Aufzeichnung

OK M=Men
```

Abb. 7.2.16 Das Bild zeigt einen gelungenen Test an

```
QUADRAT:=$
22 8900.W
21 #4.W
CD SCHLEIFE
2A 8900.W
CD SCHREITE
21 #90.W
CD DREHE
CD ENDSCHLEIFE
C9

VIELE:=$
21 #50.W
CD QUADRAT
21 #80.W
CD QUADRAT
21 #110.W
CD QUADRAT
21 #140.W
CD QUADRAT
C9
```

Abb. 7.2.17 Viele Quadrate. Beachten Sie, daß in unserer Grafik-Sprache immer zuerst die Unterprogramme kommen und dann erst das Hauptprogramm, das diese Unterprogramme benutzt. Der Start des Gesamtprogrammes muß immer beim Hauptprogramm und nicht bei der niedrigsten Programmspeicherzellennummer erfolgen

„.W“ steht für Wort. Entsprechend gibt es auch „.B“ für Byte, wenn man nur genau ein Byte, also bei uns genau eine Speicherzelle belegen will. In einem Byte lassen sich 256 verschiedene Zahlen unterbringen, in einem Wort 65536.

Jetzt benötigt man noch einen weiteren Befehl, um das ganze sinnvoll zu machen.

Der Speicher-Wert-Befehl

Im Befehl 22 xxxx.W ist xxxx wieder eine Adresse. Damit wird ein Wert, der vorher geladen oder errechnet wurde, auf die zwei Speicherplätze abgelegt, die durch xxxx bestimmt sind.

Ein vollständiges Programmbeispiel zeigt *Abb. 7.2.17*. Achtung!: Die Lade-Indirekt-Adresse ist dort sedezimal angegeben, also ohne das Zeichen „#“. *Abb. 7.2.18* zeigt das Ergebnis des Programmlaufes auf dem Bildschirm. *Abb. 7.2.19* zeigt den Speicherauszug als Kontrolle.

Hier noch ein Hinweis zu den Symbolen, also den Namen, denen man bestimmten Adressen oder Zahlen gegeben hat. Wenn man ein einzelnes Symbol löschen will, so kann man das mit folgender Anweisung tun:

SYMBOL: = %

(Anstelle von SYMBOL muß der zu löschende Name geschrieben werden.) Das Prozentzeichen ist dabei der Löschbefehl. Man verwendet ihn wie das Dollarzeichen. Will man alle Symbole löschen, so kann man den Rechner kurz ausschalten.



Abb. 7.2.18 Das Ergebnis von VIELE

```
Speicher ansehen

+=weiter -=rueckw R=Adr M=Menue

--   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
8800 22 00 89 21 04 00 C0 0F 00 2A 00 89 C0 03 00 21
     '        !        M        *        M        !
8810 5A 00 C0 06 00 C0 12 00 C9 21 32 00 C0 00 88 21
     Z     M        M        [  !  2     M        !
8820 50 00 C0 00 88 21 6E 00 C0 00 88 21 8C 00 C0 00
     P     M           !  n     M           !        M
8830 88 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        [
8840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Abb. 7.2.19 Der Speicherauszug zu Abb. 7.2.17

Wie funktioniert das Programm aus Abb. 7.2.17? Zunächst wird im Quadratprogramm mit dem Befehl „22 8900.W" ein Zahlenwert auf Adresse 8900 abgespeichert. Der Zahlenwert kann zum Beispiel durch einen normalen Ladebefehl, der vor Aufruf des Quadratprogramms ausgeführt sein muß, bestimmt werden. Dann wird mit 21 # 4.W der Schleifenzähler geladen und der Schleifen-Befehl ausgeführt. Danach wird mit dem Lade-indirekt-Befehl „2A 8900.W" der Inhalt der Speicherzelle 8900 (und 8901) wieder geladen. Nun wird der Schreite-Befehl ausgeführt.

Im Programm „VIELE", das das eigentliche Hauptprogramm ist, wird zunächst mit dem Lade-Wert-Befehl die Seitenlänge eines Quadrates geladen und dann das Programm „Quadrat" aufgerufen. Der erste Befehl findet also eine sinnvolle Zahl vor, die Seitenlänge. Er sorgt dafür, daß dieser Parameter, so nennt man eine solche von Fall zu Fall die Größe des Quadrates bestimmende Zahl, richtig ins Unterprogramm kommt. So werden vier verschiedene Quadrate ausgegeben. Man nennt das Zusammenspiel der Befehle zwischen Haupt- und Unterprogramm Parameterübergabe, da dem Unterprogramm „Quadrat" ein Parameter, die Seitenlänge des Quadrats, mitgegeben wird. Eigentlich sind Parameter ja nichts Neues mehr, denn der Schreite-, Drehe- oder Schleife-Befehl hat ja auch schon einen Parameter verwendet, nämlich die Schrittlänge, den Winkel oder die Anzahl der Schleifendurchläufe. Wichtig ist für Sie vielleicht bei unserem Sprachsystem, daß Sie fast immer die Unterprogramme vor dem Hauptprogramm im Speicher angelegt haben. Ein Programmlauf beginnt immer beim Hauptprogramm, also fast am Ende des ganzen Programmes! Jetzt kann man auch mal das Kreisprogramm als Unterprogramm verwenden. *Abb. 7.2.20* zeigt ein Programmbeispiel, *Abb. 7.2.21* das Ergebnis. Will man eine feinere Stufung haben, so kann man zum Beispiel mit 180 Schleifendurchläufen und 2 ° pro Drehung arbeiten – oder noch extremer.

Ins Innere des Z80

Der Z80 besitzt in seinem Inneren eine Reihe von speziellen Speicherzellen. Diese Speicherzellen nennt man Register. Der Z80 kann sich dort Werte merken. Der Befehl „21" lädt zum Beispiel

```
KREIS:=$
22 8900.W
21 #360.W
CD SCHLEIFE
2A 8900.W
CD SCHR16TEL
21 #1.W
CD DREHE
CD ENDSCHLEIFE
C9

KRINGEL:=$
21 #10.W
CD KREIS
21 #11.W
CD KREIS
21 #12.W
CD KREIS
21 #13.W
CD KREIS
C9
```



Abb. 7.2.21 Hier sind es vier Kreise

Abb. 7.2.20 (links) Mehrere Kreise werden gezeichnet

einen Datenwert von außen in ein bestimmtes 16-Bit-Register im Inneren des Z80. Das Register besitzt einen Namen und heißt HL-Register. Warum gerade HL? Das hat der Hersteller einfach so definiert. Register besitzen also meist Namen und keine Adressen, um sie leicht von normalen Speicherzellen unterscheiden zu können.

Der Z80 besitzt insgesamt sehr viele Register. Es sind dies neben HL noch BC, DE, HL', DE', BC', AF und AF' sowie IX, IY und SP, außerdem R und I und PC. Das Register PC ist der Programmzähler. Es bestimmt, von welcher Adresse im Speicher der nächste Befehl geholt wird. Am Anfang benötigt man noch nicht gleich alle Register, man kann sie nach und nach kennenlernen.

Der Z80 kann addieren

Zwei neue Befehle:

1. 11 xxxx.W

Der Befehlscode „11" ist der Lade-Wert-Befehl für das Register DE. Der Lade-Wert-Befehl „11" arbeitet wie der Befehl „21", nur daß der Zahlenwert nicht in HL, sondern in DE abgespeichert wird.

„Addiere DE nach HL" heißt dieser Befehl, der keine weiteren Angaben benötigt. Das Ergebnis ist anschließend im Register HL zu finden.

Beispiel:

```
21 #5.W
11 #4.W
19
```

Nach Ausführung dieses Programms steht der Wert 9 im Register HL. Ein nachfolgender Befehl „CD SCHREITE“ würde also 9 Schritte ausführen.

Dieser Addiere-Befehl ist interessant, wenn er in einer Schleife ausgeführt wird. Dazu ein Beispiel. Es soll eine Spirale gezeichnet werden.

Abb. 7.2.22 zeigt das Programm. Bei jedem Schleifendurchlauf in SPIRALE wird der Inhalt der Speicherzelle „MERKER“ (nach CD VIERTEL) um 5 erhöht. Damit wird der Viertelkreis immer größer und so ergibt sich eine Spiralform. Bei Beginn des Programms „SPIRALE“ wird zunächst der Wert 1 geladen und in „MERKER“ abgespeichert. Damit ist ein Startwert vorgegeben. Man nennt solch eine Vorbesetzung auch Initialisierung.

Beim Eingeben darauf achten, daß die Zuweisung „MERKER: = 8900“ nicht vergessen wird. Man könnte anstelle des Namens „MERKER“ auch direkt die Zahl 8900 setzen. *Abb. 7.2.23* zeigt die gezeichnete Spirale.

Ein raffiniertes Programm

Für das nächste Beispiel wird ein neuer Befehl benötigt: EB

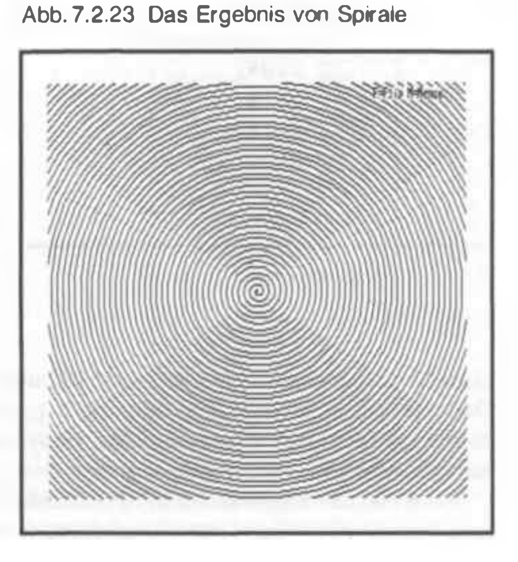
```
MERKER:=8900

VIERTEL:=$
21 #9.W
CD SCHLEIFE
2A MERKER
CD SCHR16TEL
21 #10.W
CD DREHE
CD ENDSCHLEIFE
C9

SPIRALE:=$
21 #1.W
22 MERKER
21 #256.W
CD SCHLEIFE
CD VIERTEL
2A MERKER
11 #5.W
19
22 MERKER
CD ENDSCHLEIFE
C9
```

Abb. 7.2.22 (links) Das Programm Spirale

Abb. 7.2.23 Das Ergebnis von Spirale



```
GROESSE:=8900
WACHSTUM:=8902
WINKEL:=8904
ANZAHL:=8906

UNIVERS:=$
21 #36.W
CD SCHLEIFE
2A GROESSE
22 WACHSTUM
2A ANZAHL
CD SCHLEIFE
2A GROESSE
CD SCHREITE
2A WINKEL
CD DREHE
2A WACHSTUM
EB
2A GROESSE
19
22 GROESSE
CD ENDSCHLEIFE
2A WACHSTUM
22 GROESSE
CD ENDSCHLEIFE
C9

SPIRO:=$
21 #10.W
22 GROESSE
21 #120.W
22 WINKEL
21 #7.W
22 ANZAHL
CD UNIVERS
C9
```

Abb. 7.2.24 Dieses Programm arbeitet sehr intensiv mit Parametern



Abb. 7.2.25 Ein kleines Kunstwerk, gefertigt vom Programm aus Abb. 7.2.24

Damit wird der Inhalt der Registerpaare DE und HL vertauscht. *Abb. 7.2.24* zeigt das Programm. Darin wird sehr intensiv mit Parametern gearbeitet. Im Programmteil „SPIRO“ werden die Startwerte festgelegt. Dann wird das Unterprogramm „UNIVERS“ aufgerufen. *Abb. 7.2.25* zeigt, was passiert, wenn man es startet, *Abb. 7.2.26* den Speicherauszug.

Wenn man die Parameter in SPIRO abändert, kann man andere Bilder erzeugen. Wenn man zum Beispiel den Winkel mit 60 vorbelegt, so entsteht *Abb. 7.2.27*. Wenn man den Winkel mit



Abb. 7.2.26 Das ist der Speicherauszug

Abb. 7.2.27 Nur ein Parameter in Spiro wurde geändert

190 und die GROESSE mit 20 vorbelegt, so entsteht *Abb. 7.2.28*. Wie funktioniert das Programm? *Abb. 7.2.29* zeigt ein Struktogramm des Unterprogramms „UNIVERS".

Das Programm besteht aus zwei ineinandergeschachtelten Schleifen. Die äußere Schleife wird hier 36mal durchlaufen. Diesen Wert kann man auch abändern. Dann ändert sich die gezeichnete Figur.

Die innere Schleife wird ANZAHL mal durchlaufen. Man hätte hier auch einen LADE-WERT-BEFEHL „21" verwenden und die Anzahl direkt laden können. Bei GROESSE und WACHSTUM wäre das nicht möglich, denn sie werden während eines Programmlaufes immer wieder



Abb. 7.2.28 Noch eine Parameteränderung

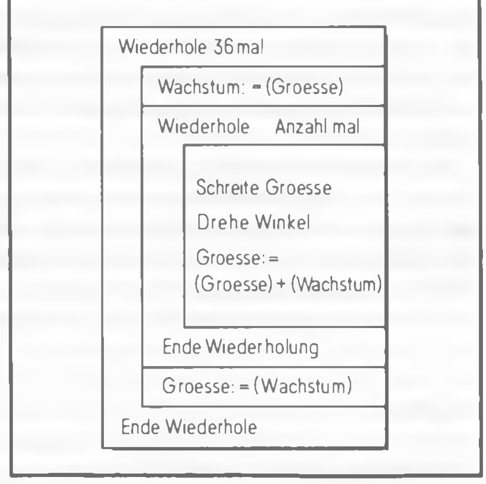


Abb. 7.2.29 Das Struktogramm des Programms Univers aus Abb. 7.2.24

21 xxxx.W	Lade-Wert-Befehl nach HL
2A xxxx.W	Lade-Indirekt-Befehl nach HL
22 xxxx.W	Speichere-Wert-Befehl von HL
11 xxxx.W	Lade-Wert-Befehl nach DE
19	Addiere DE nach HL
EB	vertausche HL mit DE
CD xxxx.W	Rufe Unterprogramm xxxx.W auf

Anstelle von xxxx.W kann auch ein Name stehen.

Abb. 7.2.30 Das sind die bisher bekannten Befehle

abgeändert. In der Schleife wird zunächst der Inhalt der Speicherzelle GROESSE in die Speicherzelle bei WACHSTUM gelegt. Im Struktogramm ist es die Zeile:

WACHSTUM : = (GROESSE)

Die Klammern bedeuten, daß der Lade-Indirekt-Befehl verwendet wird. Denn es wird der Inhalt der Speicherzelle GROESSE und nicht die Adresse nach WACHSTUM gegeben. Dann folgt die innere Schleife. Dort wird um GROESSE weitergeschritten und um WINKEL gedreht. Dann folgt eine Addition. Es wird der Inhalt von GROESSE zum Inhalt von WINKEL addiert und in GROESSE abgespeichert. Nach Beenden der Schleife wird der Inhalt von WACHSTUM wieder in GROESSE gespeichert. Damit besitzt GROESSE den gleichen Inhalt wie vor dem Start der Schleife.

Hier noch eine Bemerkung zur Zählschleife, wie sie auch im vorherigen Programmbeispiel vorkam. Mit einer Zuweisung in einem Struktogramm, wie MERKER : = (MERKER) + 5, wird der Inhalt der Speicherzellen bei der Adresse MERKER um 5 erhöht. Führt man diesen Befehl in einer Schleife aus, so wächst die Zahl bei jedem Durchlauf um 5. Man sagt in diesem Fall auch, daß es sich um einen Zähler handelt.

Abb. 7.2.30 zeigt die Befehle, die bis jetzt besprochen worden sind.

Aufgaben

1. Was tut der Befehl 11 #10.W?
2. Wieviel Schritte werden nach Ausführung des Schreitebefehls beim Programm aus *Abb. 7.2.31* durchlaufen?

Abb. 7.2.31 Programm zu Aufgabe 2

```
21 #50.W
11 #10.W
EB
CD SCHREITE
C9
```

3. Entwerfen Sie ein Programm, das Quadrate mit der Seitenlänge 1 bis zur Seitenlänge 400 ausgibt?
4. Man entwerfe ein Programm, das Blumen in unterschiedlicher Größe auf dem Bildschirm ausgibt (BLUMENGARTEN).

7.3 Serielles Interface

Als nächstes soll eine Karte mit einem seriellen Interface besprochen werden. Danach ist es z. B. möglich, ein handelsübliches Datensichtgerät oder eine Datensichtgerätekarte an den Computer anzuschließen. Ein solches Gerät ist für das weitere Arbeiten mit unserem Computer aber nicht zwingend notwendig.

Zunächst etwas über die Grundlagen der seriellen Übertragung. Wir haben gelernt, daß die Daten seriell, das heißt zeitlich aufeinanderfolgend über eine Leitung übertragen werden können. Wie aber geschieht dies nun genau?

Es gibt dafür in der Praxis eine Vielzahl von Möglichkeiten; wir wollen aber nur die gebräuchlichste verwenden.

Soll ein 8-Bit-Datenwort übertragen werden, so wird dies in einzelne Bits zerlegt, die dann eins nach dem anderen auf die serielle Leitung geschaltet werden. Beim Empfänger müssen die einzelnen Bits wieder zu einem Datenwort zusammengesetzt werden. Dazu ist es aber nötig, zu wissen, wann das erste Bit übertragen wurde, und wie lang jeweils ein Bit auf der Leitung ist. Dafür gibt es ein Standardformat. In *Abb. 7.3.1* ist das Format abgebildet. Übertragen wird mit einer festen Bitrate, die auch als Baudrate bezeichnet wird. Diese Zahl gibt an, wie viele Bits in einer Sekunde übertragen werden. Benötigen die einzelnen Bits eine Breite von 3.3 ms, so ergibt sich eine Baudrate von 300 Baud. Um dem Empfänger mitzuteilen, wann die Übertragung stattfindet, gibt es am Anfang ein sogenanntes Startbit. Dieses Startbit trägt keine Dateninformation und dient nur dazu, dem Empfänger den Beginn einer Datenübertragung mitzuteilen. Danach folgen in unserem Fall die Datenbits von 0 bis 7. Begonnen wird mit dem niederwertigsten Datenbit. Nach den Daten folgen Stop-Bits, um einen Abstand zwischen dieser Übertragung und einer eventuell gleich anschließenden Übertragung zu garantieren. Die Anzahl der Datenbits reicht von 5 Bits, bei alten Fernschreibern über 7 Bits, um eine Textübertragung mit dem ASCII-Satz zu ermöglichen, bis zu 8 Bits bei Datenübertragungen von vollständigen Bytes. Es kann nach den Datenbits auch noch ein Paritätsbit folgen, das zusätzlich übertragen wird. Dieses Paritätsbit ist die Quersumme über die Datenbits: Damit kann beim Empfänger geprüft werden, ob ein Fehler bei der Übertragung vorlag.



Abb. 7.3.1 Format eines seriellen Bitstromes

Beispiel GERADE (even) Parität bei 7 Datenbits + 1 Paritätsbit

1 0 1 1 0 1 1 1

Oder bei UNGERADER (odd) Parität

1 1 0 1 0 1 0 1

Gerade oder ungerade Parität wird nach Konvention verwendet und muß im Sender und Empfänger übereinstimmen. Werden 8 Datenbits und Parität übertragen, so sind 9 Bits über die Leitung zu transportieren, wobei ein Startbit und ein oder zwei Stop-Bits hinzukommen. Es können ein oder zwei Stop-Bits ebenfalls per Konvention gewählt werden. Die Gesamtübertragungsrate errechnet sich dann wie folgt:

f/(1*START + n*DATEN + k*PARITÄT + 1*STOP)

Beispiel:
8 Datenbits keine Parität 1 Stop Bit, 1200 BAUD

1200/(1 + 8 + 0 + 1) Zeichen/sek

= 120 Zeichen/sek

Es können dann 120 Zeichen pro Sekunde übertragen werden.

Wie kann z. B. der Sender aussehen? *Abb. 7.3.2* zeigt ein Schema. Es wird ein 11 Bit-Schieberegister verwendet. Das Schieberegister erhält über den Takteingang direkt die Baudrate, also bei 1200 Baud entsprechend 1200 Hz. Die zu übertragenden Daten werden an die Paralleleingänge B bis I angeschlossen, hier sollen 8 Datenbits, keine Parität und zwei Stop-Bits verwendet werden. Das Start-Bit wird durch eine Massenverbindung an den Eingang A erzeugt und die beiden Stop-Bits werden durch J, K und + 5 V erzeugt. Wird nun ein Ladepuls an den

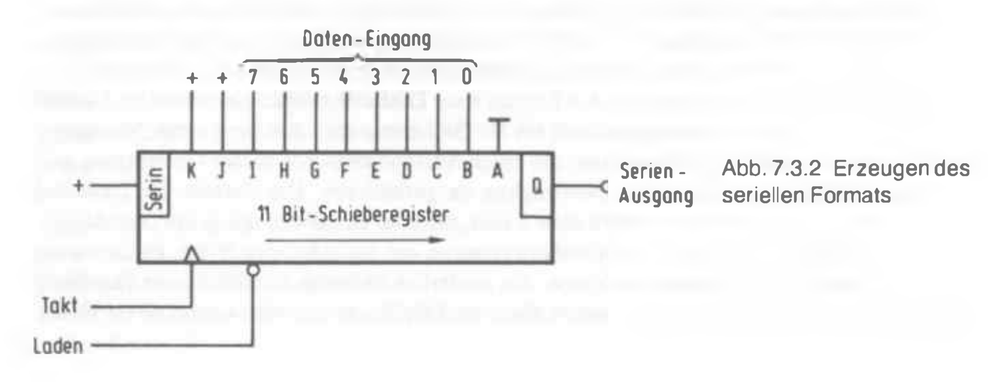


Abb. 7.3.2 Erzeugen des seriellen Formats

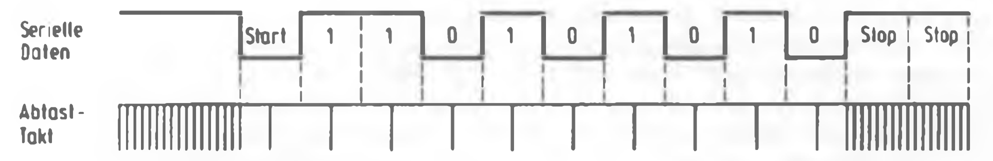


Abb. 7.3.3 Abtastung des Signals beim Empfänger

Lade-Eingang des Schieberregisters gelegt, so beginnt die Übertragung. Nach beendeter Übertragung befinden sich im Schieberegister lauter 1-Werte, die über den Eingang Ser des Schieberegisters laufend nachgeschoben werden. Soll ein neues Datenwort übertragen werden, so muß mit einem externen Zähler festgestellt werden, ob die Übertragung des vorherigen Datenwortes schon beendet ist. Bei einem Ladevorgang des Schieberegisters kann dieser externe Zähler z. B. anfangen, bis auf 11 zu zählen. Ist der Wert erreicht, so kann ein neuer Ladevorgang stattfinden, der auch den Zähler wieder rücksetzen müßte.

Wie kann das serielle Signal wieder zurückgewonnen werden? Dazu *Abb. 7.3.3*. Im oberen Bildteil ist das serielle Signal aufgetragen, unten der sogenannte Abtasttakt. Mit dem Abtasttakt wird jeweils der Wert auf der Datenleitung in ein internes Register übernommen. Der Abtasttakt kann z. B. der 16fache Wert der Baudrate sein. Nun beginnt der Vorgang: Sobald das Startsignal erscheint, also eine 0 auf der Leitung liegt, wird der Abtasttakt umgeschaltet, als erstes wird nun nach der Zeit t/2 eine erneute Abtastung durchgeführt, ist dort die Leitung immer noch auf 0, so liegt tatsächlich ein Startsignal vor. Wenn nicht, wird alles ignoriert und die Abtastung mit t/16 fortgesetzt. Ist aber das Startsignal da, so kann nach der Zeit t erneut abgetastet werden, und das erste Datenbit kann in ein Schieberegister geschoben werden. Die restlichen Datenbits oder ggf. Paritätsbits werden ebenfalls in das Schieberegister eingetragen. Nun folgen die Stop-Bits. Dort wird geprüft, ob die Leitung auf 1 liegt, tut sie das nicht, sondern liegt sie noch auf 0, so wurde ein Übertragungsfehler erkannt, der auch den Namen FRAMING ERROR besitzt. Er tritt auch auf, wenn z. B. im Sender die Anzahl der Datenbits + Paritätsbit größer als die der im Empfänger eingestellten ist. Das Übertragungsverfahren wird auch als asynchrones Übertragungsverfahren bezeichnet, da Sende- und Empfangstakt nicht starr miteinander verbunden sind. Es gibt auch ein synchrones Übertragungsverfahren, das aber die zusätzliche Übertragung der Taktfrequenz benötigt, die entweder über eine getrennte Leitung oder durch ein Modulationsverfahren mit den Daten übertragen werden muß. Mit synchroner Übertragung können im allgemeinen höhere Baudraten verwendet werden, doch ist das Verfahren aufwendiger und für unsere Zwecke nicht von Nutzen.

Zum Glück müssen wir aber Sender und Empfänger nicht selbst aus einzelnen Gattern zusammenbauen, denn es gibt fertig integrierte Bausteine, die all dies beinhalten. Da ist eine Vielzahl dieser Bausteine mit unterschiedlichen Eigenschaften. Aus dieser Vielzahl heraus verwenden wir einen besonders komfortablen, den Baustein 6551. *Abb. 7.3.4* zeigt die gesamte Schaltung des seriellen Interface. Hier wird nun erstmals eine Einheit nicht über den Speicheradressenraum angesprochen, sondern über die Peripherieadressen adressiert.

Der Datenbus ist mit dem bidirektionalen Bustreiber 74LS245 gepuffert. Der Richtungseingang Dir ist dabei wie auch bei den Speicherschaltungen mit -RD verbunden, denn bei einem Lesezugriff soll die Datenrichtung von der Karte auf den Bus zeigen. Der Baustein B1 wird immer dann freigegeben, wenn einmal das Signal -IORQ vorliegt, da es sich um einen Peripheriezugriff handeln soll; zum andern muß die Karte adressiert sein. Die Auswahl des Adreßbereichs geschieht mit Hilfe des Vergleichers 74LS85 (IC7 und IC8). Der Baustein 6551 benötigt selbst zwei Adreßleitungen für vier interne Register, so daß die Adressen A0 und A1 reserviert sind. Die Adressen A2 bis A7 können dagegen zur Adressierung hinzugezogen werden. A8 bis A15 sind

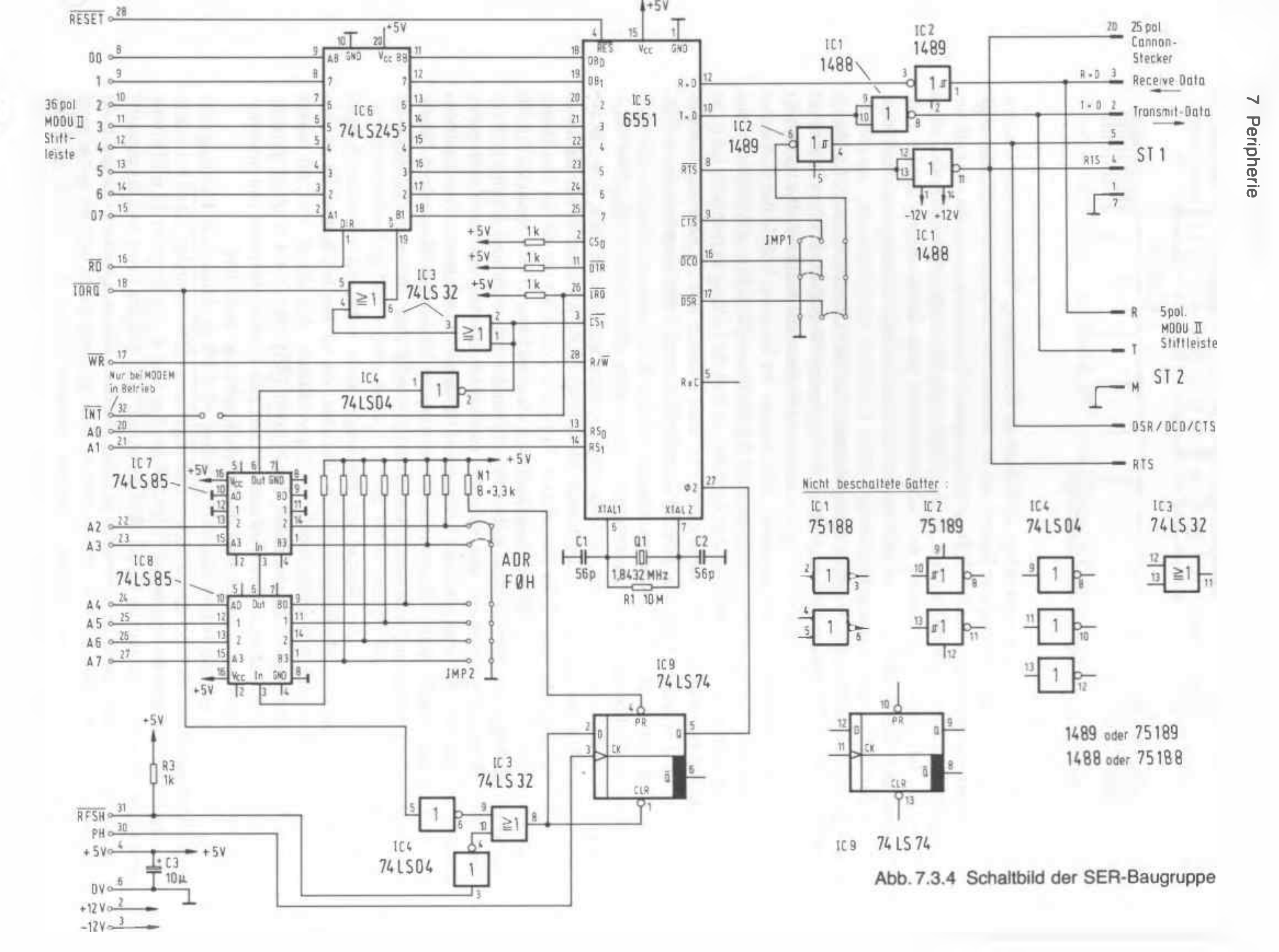


Abb. 7.3.4 Schaltbild der SER-Baugruppe

hier ohne Bedeutung, da bei der Adressierung von Peripheriegeräten beim Z80 nur 256 Adressen verwendet werden. A2 bis A7 gelangen daher an die Vergleicher. An den gegenüberliegenden Eingängen der Vergleicher kann mit Hilfe von Brücken eine Vergleichsadresse eingestellt werden. Wird keine Brücke nach Masse eingesetzt, so reagiert die Karte auf die Adressen FCh, FDh, FEh und FFh. Am Gleich-Ausgang von IC7 erscheint immer dann ein 1-Signal, wenn die Adressen von A2 bis A7 mit der Vergleichsadresse übereinstimmen. Das ist sowohl bei Peripherie- als auch bei Speicherzugriffen der Fall. Daher wird über IC3 (Pin 4, 5, 6) die Verknüpfung mit -IORQ durchgeführt; nun reagiert die Schaltung nur noch auf Peripherieadressen. Der Ausgang hinter IC4, Pin 2 wird aber direkt an den Eingang -CS1 des seriellen Bausteins geführt, und die Verknüpfung mit -IORQ geschieht dort indirekt über die Leitung PHI2 (Pin 27). Das Signal -WR wird direkt an den Eingang R/-W des Bausteins 6551 geführt und gibt an, wann ein Schreibzugriff vorliegt. Das Signal PHI2 ist leider nicht direkt mit einem Signal des Z80-Busses verbindbar, da der Baustein urpsprünglich für 6502-Systeme entworfen wurde. Eine kleine Anpaßschaltung, bestehend aus IC9, IC4 (5 u. 6), IC3 (9, 8, 10) und IC4 (4 und 3), erzeugt das passende Signal. Über IC4 wird auch das -IORQ-Signal wieder in die Schaltung eingeführt. Das Signal -RFSH wird hier nicht zum Refresh von Speichern benötigt, erfüllt aber eine ähnliche Aufgabe. Es liegt permanent vor, und zwar immer zwischen I/O-Zugriffen. Durch die Schaltung zur Erzeugung von PHI2 wird ein nahezu kontinuierlicher Takt für den Baustein 6551 erzeugt, der aber bei Zugriffen auf den Baustein das Signal -IORQ trägt.

Der serielle Baustein benötigt einen eigenen Quarz, der die Frequenz 1.8432 MHz erzeugen muß. Diese krumme Frequenz wird benötigt, um intern standardisierte Baudraten erzeugen zu können. Die Baudrate ist im übrigen programmierbar.

In unserer Schaltung kommen neben dem seriellen Ausgang TxD und dem seriellen Eingang RxD auch noch weitere Ein- und Ausgänge vor. Der Eingang -DSR kann eine Übertragung verhindern, wenn er auf einem 1-Signal liegt (Pin 9). Damit ist es möglich, außen ein langsames Gerät anzuschließen, das in der Lage ist, die Übertragung für kurze Zeit anzuhalten, um die eingetroffenen Daten zu verarbeiten. Umgekehrt kann der Baustein über die Leitung -RTS verlangen, daß die Übertragung bei dem anderen Gerät gestoppt wird. Die Leitung wird dazu mit dem -DSR-Eingang des sendenden Gerätes verbunden. Natürlich muß die Software beider Geräte für einen solchen Quittungsbetrieb ausgelegt sein.

In den besprochenen Leitungen liegt der Treiberbaustein IC1. Er hat die Aufgabe, den Pegel von 0 V bis 5 V auf ein Niveau + 12 V bis −12 V anzuheben. Auf diesen Spannungspegel hat man sich geeinigt, und er garantiert die Übertragung auch über große Entfernungen. Die Bausteine 75188 (1488) und 75189 (1489) übernehmen die Pegelwandlung. Der Sender 75188 wird dazu mit +/− 12 V versorgt. Beim Empfänger ist dies nicht nötig. Das Signal -RTS kann per Software aktiviert werden. Der Baustein besitzt noch eine Reihe weiterer Ein- und Ausgänge, wie -DTR, -DCD, -CTS, die aber für uns hier nicht weiter von Bedeutung sind.

Der Ein-Ausgang RxC kann eine externe Baudrate aufnehmen oder dient als Ausgang der internen, je nach Programmierung des seriellen Bausteins. Der Ausgang -IRQ ist ein Interrupt-Ausgang, den wir jedoch ebenfalls nicht verwenden werden.

Der Baustein besitzt – wie schon erwähnt – mehrere interne Register. *Abb. 7.3.5* zeigt ihre Bedeutung. Wird das Register 0 angesprochen, so wird das Datenregister aktiviert. Register 0 bedeutet in unserem Fall aber die Adresse 0FCh, wenn die Brücken am Vergleicher offen sind. Wird in das Register 0 etwas geschrieben, so werden die Daten in das Senderegister übertragen und seriell über die Leitung geschickt. Beim Empfang kommen die Daten im Empfangsregister an, das ebenfalls über die Adresse 0FCh erreichbar ist, diesmal aber bei einem Lese-Zugriff. Nun genügen diese Register allein natürlich nicht, um eine Datenübertragung durchführen zu können. Zum einen müssen wir wissen, ob die abgesandten Daten vollständig aus dem Senderegister übertragen wurden, um einen nächsten Datenwert eingeben zu können, und dann müssen wir

RS1	RS0	Schreiben	Lesen
0	0	Sende Daten	Empfangsdaten
0	1	Software Reset	Status Register
1	0	Command Register	
1	1	Control Register	

Abb. 7.3.5 Registrierbelegung beim Baustein 6551

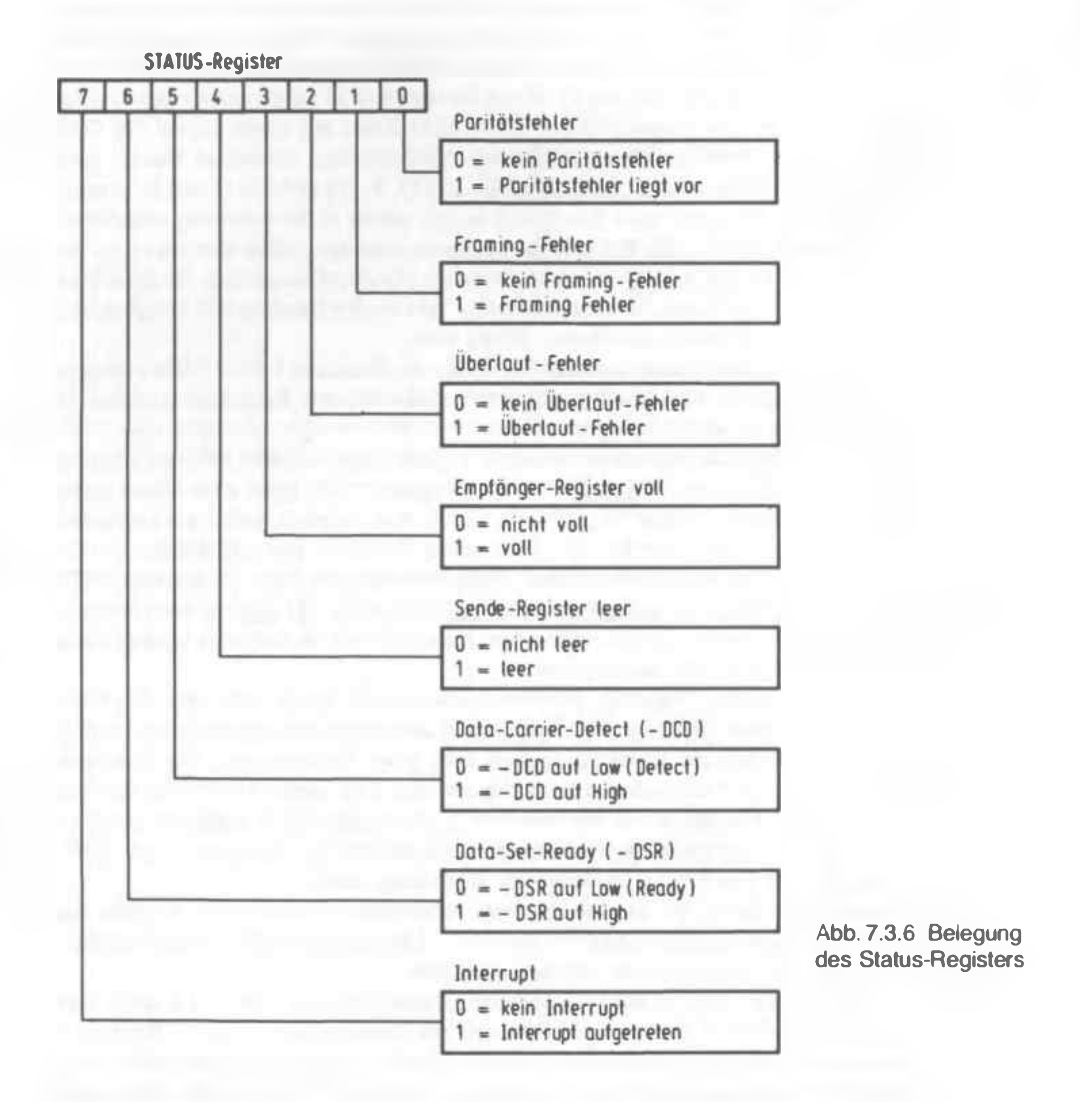


Abb. 7.3.6 Belegung des Status-Registers

wissen, ob Daten und Empfangsteil angekommen sind. Dies ist mit Hilfe des Status-Registers möglich, das über die Adresse 0FDh (Register 1) ausgelesen werden kann. *Abb. 7.3.6* zeigt die Bedeutung der einzelnen Bits im Status-Register. Für uns sind zunächst die beiden Bits 4 und 3 interessant, die anderen beschreiben den Zustand bei Fehlern oder den Zustand von den Spezialleitungen – wie -DSR und -DCD – und Interruptmeldungen. Bit 4 ist genau dann auf 1,

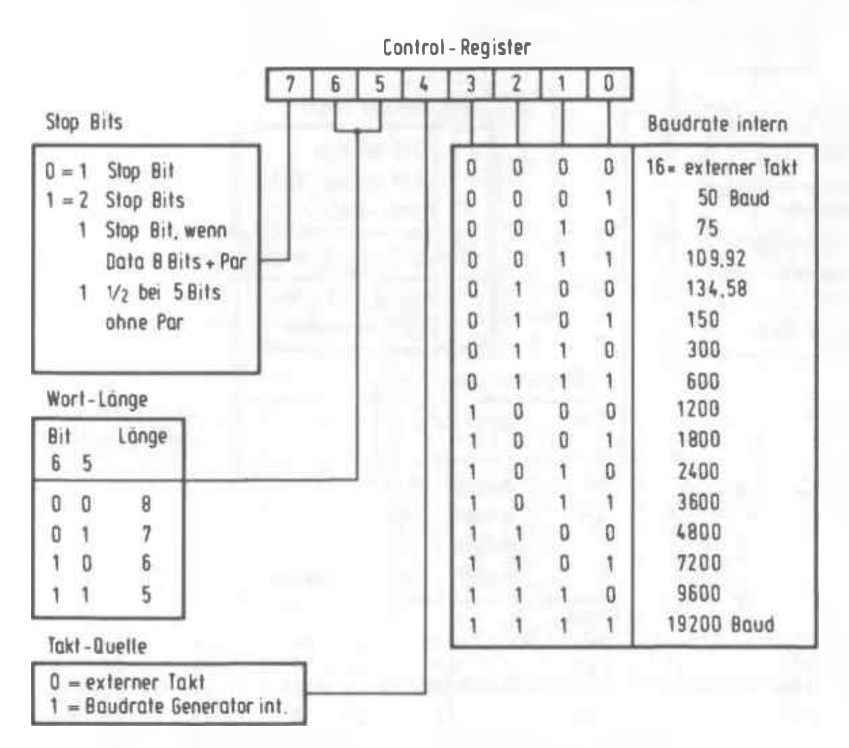


Abb. 7.3.7 Control-Register

wenn das Senderegister leer ist und ein neues Zeichen in das Register 0 geschrieben werden kann, welches dann über die Leitung übertragen wird. Bit 3 des Status-Registers ist genau dann auf 1, wenn ein Datenwort empfangen wurde und in Register 0 bereitsteht. Wird Register 0 ausgelesen, so geht dieses Bit solange wieder auf 0 zurück, bis ein neues Datenwort empfangen wurde. Wird in Register 1 geschrieben, so wird der serielle Port 6551 rückgesetzt, ähnlich, als ob er einen physikalischen Reset über die Leitung-RESET empfangen hätte. Dabei ist es egal, welcher Wert in Register 1 geschrieben wurde. Die restlichen Register dienen der Voreinstellung von Parametern und der Betriebsart des Bausteins. *Abb. 7.3.7* zeigt die Bedeutung der Bits des Control-Registers 3 (Adresse 0F3h bei uns). Dort wird die Baudrate programmiert, die Wortbreite eingestellt und die Anzahl der Stop-Bits bestimmt. Wird eine Baudrate von 9600 Baud verwendet und 8 Bits mit einem Stop-Bit übertragen, so ergibt sich das Befehlsbyte: 0001 1 1 1 0b oder 1Eh.

Im Command-Register *Abb. 7.3.8* kann bestimmt werden, ob ein Paritätbit verwendet wird und wenn ja, welcher Art es sein soll. Die Begriffe Even und Odd haben wir schon kennengelernt, Mark bedeutet, daß ein fester Wert 1 angenommen wird, und Space, daß der Wert 0 verwendet wird. Dies ist keine echte Parität, sondern es wird nur das eine Bit bei der Übertragung mit verwendet. Die restlichen Bits dienen der Einstellung von -RTS, von Interrupts und der Freigabe der Übertragungskanäle. Wir verwenden keine Parität und -RTS und -DTR liegen auf Low. Damit ergibt sich als Steuerwort 00001011b oder 0Bh. Es wird außerdem der Normal-Mode mit Bit 4 eingestellt, da der Echo-Mode hier nicht verwertbar ist.

Wir hatten vorher schon einmal kurz den Begriff ASCII verwendet, hier soll erklärt werden, was darunter verstanden wird. *Abb. 7.3.9* zeigt eine Umrechnungstabelle. Jedem darstellbaren Zeichen des ASCII-Satzes (nach ISO-Norm, DIN 66003) ist ein Wert zugeordnet, der den Code

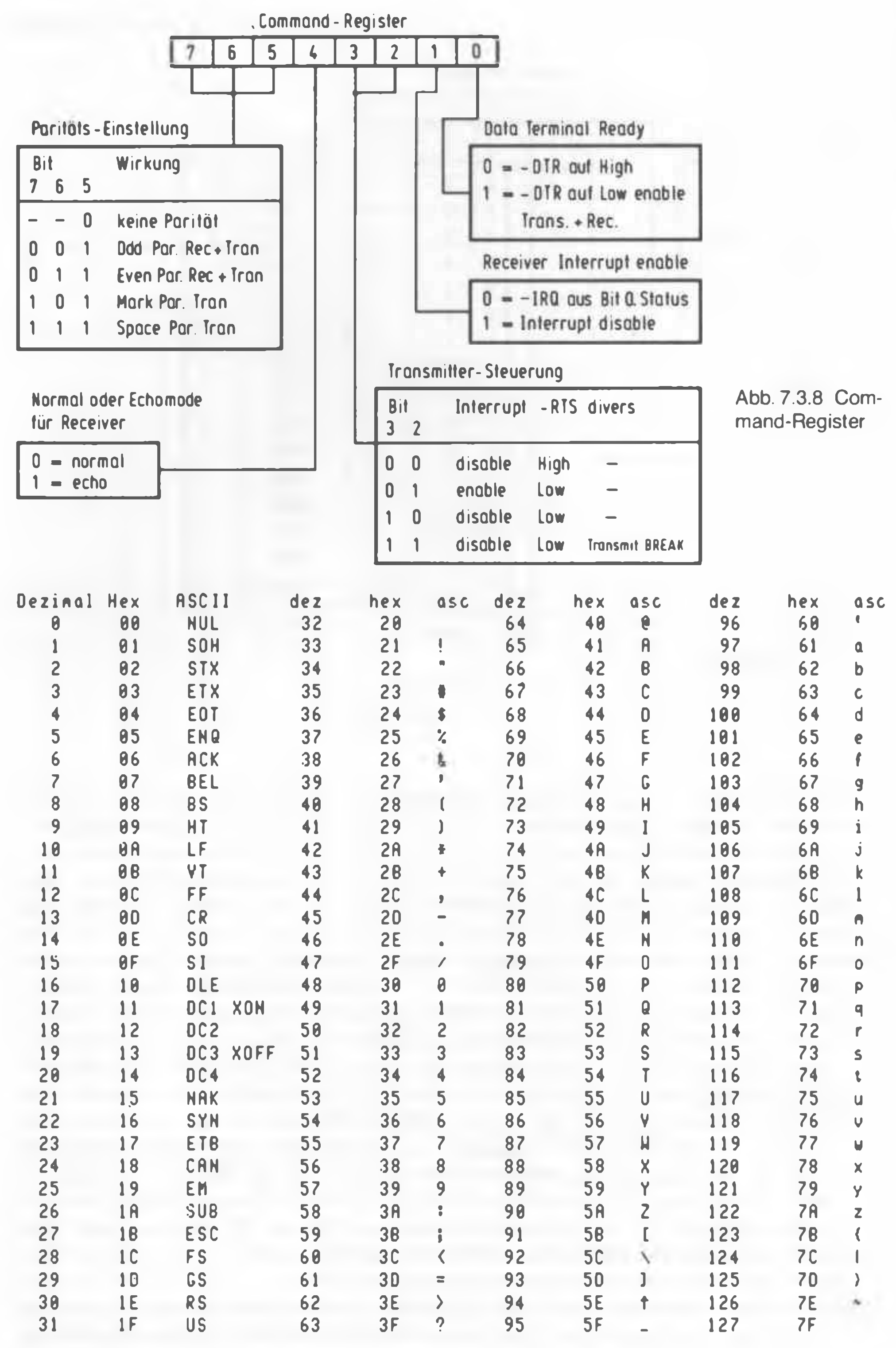


Abb. 7.3.8 Command-Register

Dezimal	Hex	ASCII	dez	hex	asc	dez	hex	asc	dez	hex	asc
0	00	NUL	32	20		64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1 XON	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3 XOFF	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	↑	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	

Abb. 7.3.9 Der ASCII-Satz

BAL.

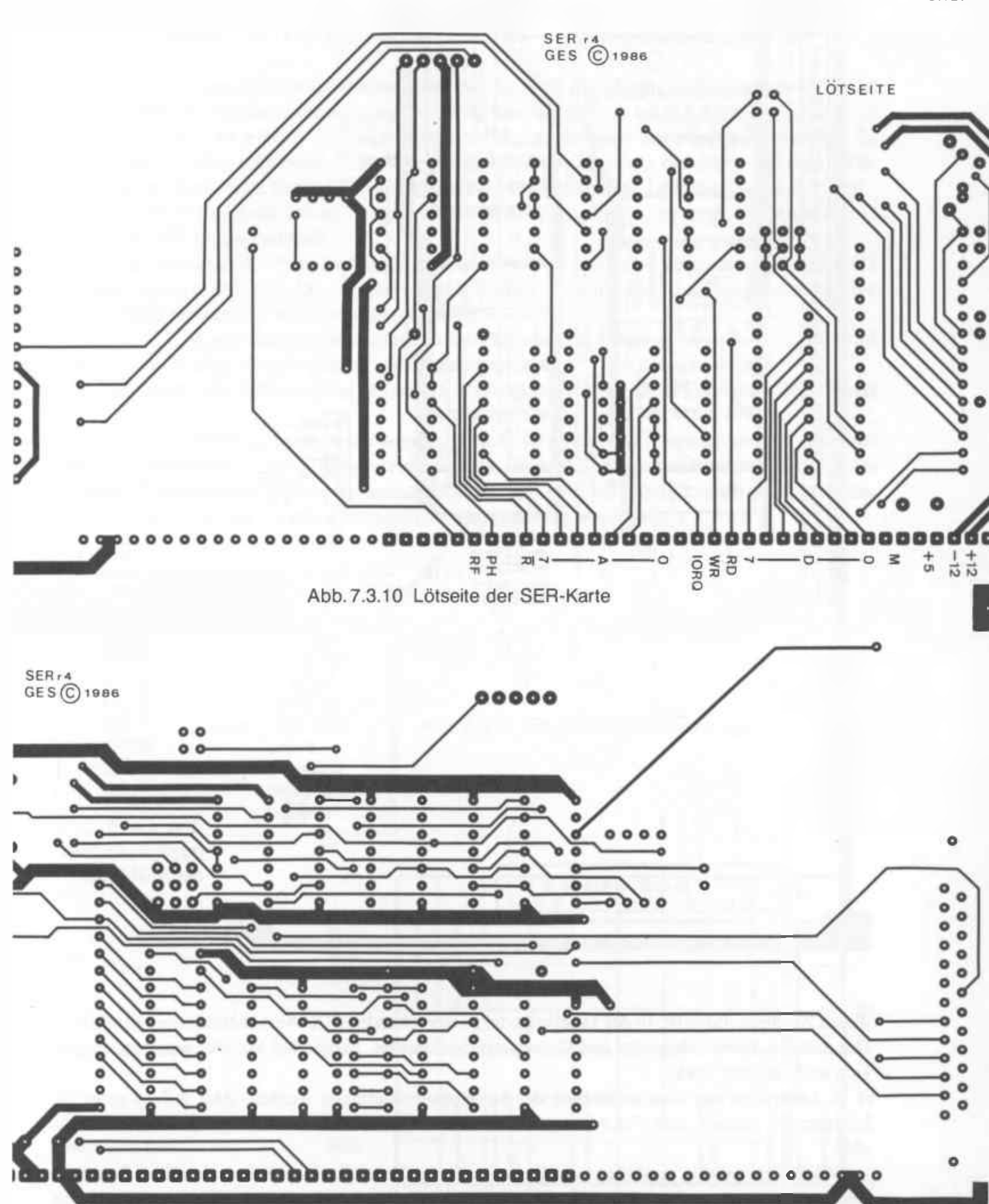


Abb. 7.3.10 Lötseite der SER-Karte

Abb. 7.3.11 Bestückungsseite der SER-Karte

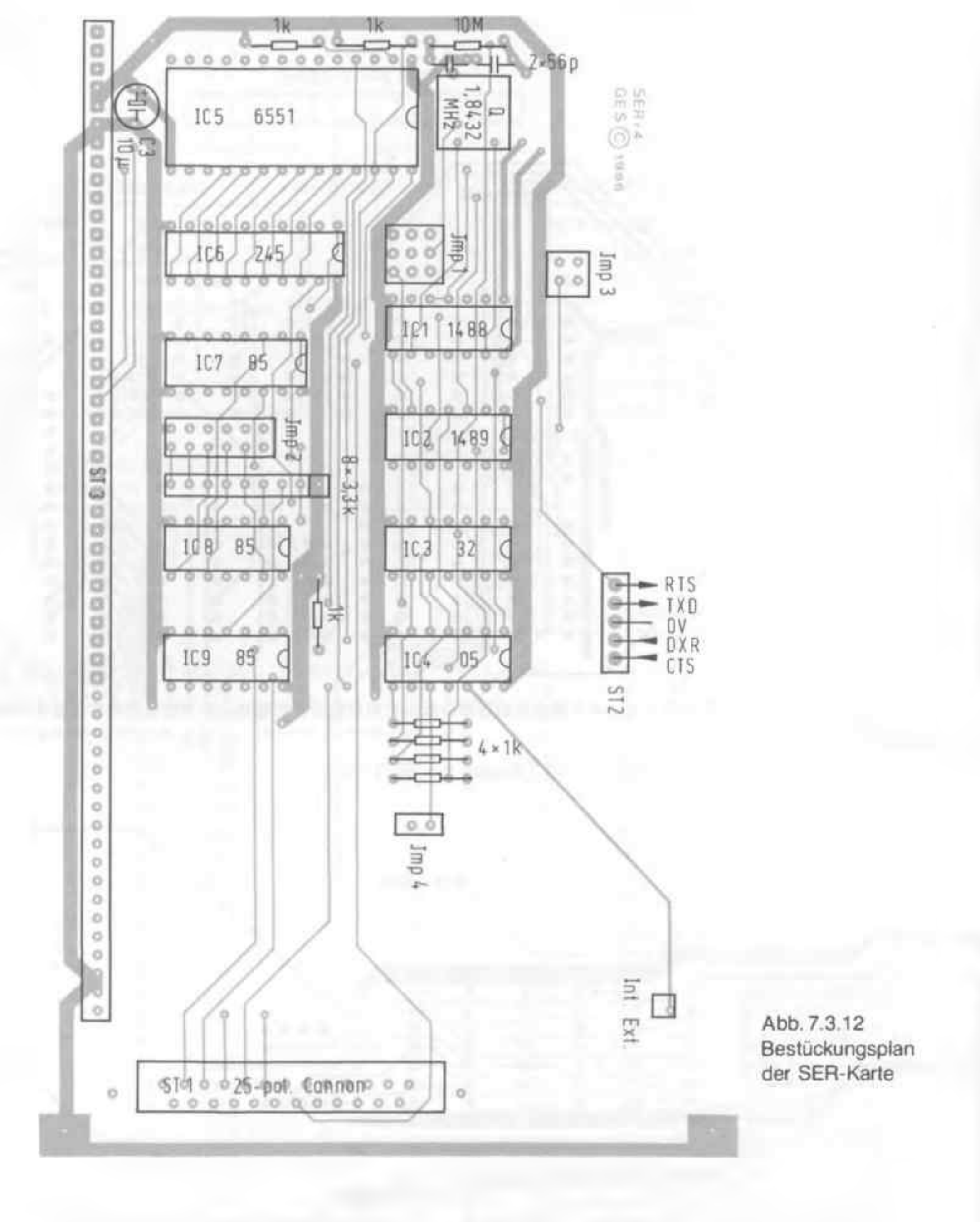


Abb. 7.3.12
Bestückungsplan
der SER-Karte

dieses Zeichens darstellt. In der Tabelle ist der entsprechende HEX-Wert ebenfalls mit abgebildet. Die meisten Datensichtgeräte und Computerschnittstellen, gerade bei Mikrocomputern, richten sich nach diesem Code.

Es kann jetzt mit dem Bestücken der Leiterplatte begonnen werden. *Abb. 7.3.10* zeigt die Lötseite der Platine, *Abb. 7.3.11* zeigt die Bestückungsseite und *Abb. 7.3.12* den Bestückungsplan.

Wie üblich werden zunächst nur alle Sockel eingelötet, die ICs bleiben draußen. Ebenfalls werden die passiven Bauteile, z. B. Quarz, Widerstände und Kondensatoren eingelötet. Dann kann der Test beginnen.

1. Messen der Versorgungsspannungen an den ICs. Als erstes messen wir alle 5 V Spannungen an den ICs, dann die +/− 12 V Spannung am IC 75188. + 12 V liegt dabei an Pin 14 und − 12 V an Pin 1.

2. Die Karte kann nun entweder zusammen mit der SBC 2 oder mit der Vollausbau-CPU und mit dem Grundprogramm betrieben werden. Für den Test genügt es, das Grundprogramm zu verwenden, oder es werden die nachfolgenden Programmstücke zum Testen verwendet. Es werden nun alle ICs eingesetzt. Eine Verbindung des DSR-Eingangs der Karte mit dem RTS-Ausgang der Karte wird hergestellt, um später zu garantieren, daß -DSR am IC auf 0 V liegt.

3. *Abb. 7.3.13* zeigt ein kurzes Testprogramm. Es definiert alle Parameter des Serienports und durchläuft eine Eingabeschleife.

4. Nach dem Start des Testprogramms muß an PIN 5 des 6551 eine Frequenz von 153600 Hz anliegen (Periode 6.51 µs). Dies ist die 16fache Baudrate, da für die Empfängersteuerung eine höhere Taktrate benötigt wird, wie schon gezeigt.

5. *Abb. 7.3.14* zeigt das Oszillogramm, das den Dekodierungsvorgang darstellt. Der 6551 wird zyklisch durch unser Testprogramm angesprochen.

6. Eine Reihe von Impulsen sind in *Abb. 7.3.15* dargestellt. Dies ist die Erzeugung des Signals PHI2.

7. Nun kann der Senderteil getestet werden. *Abb. 7.3.16* zeigt das Testprogramm, das diesmal eine Ausgabeschleife darstellt, die kontinuierlich den Wert 6Ah auf die Datenleitung gibt oder das Zeichen „j“ gemäß seiner ASCII-Bedeutung. In *Abb. 7.3.17* ist ein Oszillogramm dargestellt, das direkt am Ausgang der seriellen Schnittstelle abgenommen wurde.

Testprogramm: Empfänger

```
3E 1E            ld a,1eh
D3 F3            out (0f3),a
3E 0B            ld a,0bh
D3 F2            out (0f2h),a
DB F1    warte:  in a,(0f1h)
E6 08            and 8
28 FA            jr z,warte
DB F0            in a,(0f0h)
18 F6            jr warte
```

Abb. 7.3.13 Testprogramm: Empfänger

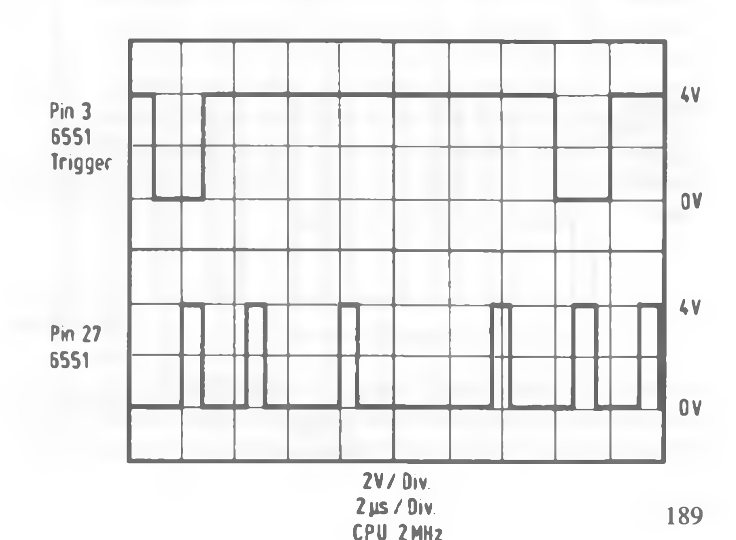


Abb. 7.3.14 Oszillogramm der Karte SER

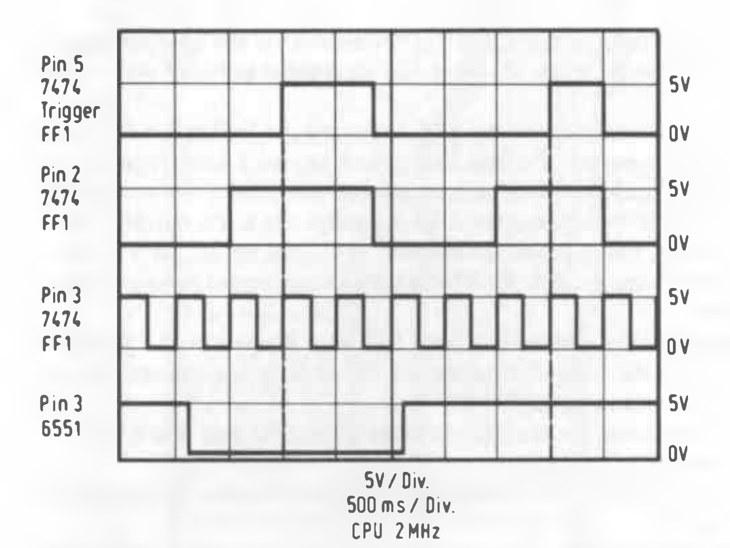


Abb. 7.3.15 Oszillogramm der Karte SER

Testprogramm: Sender		
3E 1E		ld a,1eh
D3 F3		out (0f3),a
3E 08		ld a,8
D3 F2		out (0f2h),a
DB F1	warte:	in a,(0f1h)
E6 10		and 10h
28 FA		jr z,warte
3E 6A		ld a,'j'
D3 F0		out (0f0h),a
18 F4		jr warte

Abb. 7.3.16 Testprogramm: Sender

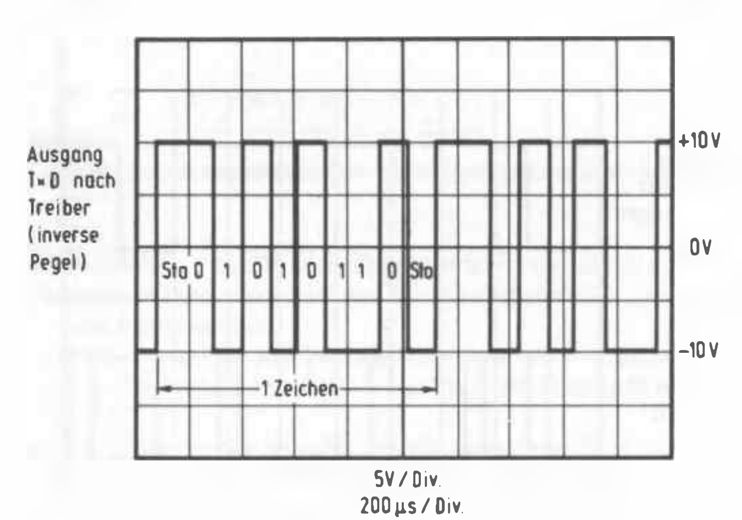


Abb. 7.3.17 Oszillogramm der Karte SER

Testprogramm: Sender/Empf.		
3E 1E		ld a,1eh
D3 F3		out (0f3),a
3E 0B		ld a,0bh
D3 F2		out (0f2h),a
0E 6A		ld c,'j'
DB F1	warte:	in a,(0f1h)
E6 10		and 10h
28 FA		jr z,warte
79		ld a,c
D3 F0		out (0f0h),a
DB F1	warte2:	in a,(0f1h)
E6 08		and 8
28 FA		jr z,warte2
DB F0		in a,(0f0h)
4F		ld c,a
18 EC		jr warte

Abb. 7.3.18 Testprogramm: Sender/Empfänger

Hier ist ferner zu beachten, daß die Treiber die Signale umkehren, daher rührt auch das inverse Bild in *Abb. 7.3.17*. Am 6551 Pin 10 ist es genau invertiert und natürlich nur mit einer 5 V (4 V) Amplitude vorhanden.

8. Auf einem Datensichtgeräteschirm müssen lauter Zeichen „j" erscheinen. Wichtig ist, daß das Datensichtgerät vor dem Stromeinschalten mit dem Rechner verbunden wurde, denn sonst kann eine fehlerhafte Synchronisation auch zu falschen Zeichen auf dem Schirm führen.

9. In *Abb. 7.3.18* ist ein weiteres Programm abgebildet. Um es zu verwenden, wird entweder ein Datensichtgerät angeschlossen, oder die Verbindung vom Eingang des Seriellen Interface zum Ausgang des seriellen Interface hergestellt. Wird nach Herstellung der Rückkopplung der RESET ausgelöst, so müssen fortlaufend 6Ah-Werte über die Leitung gesendet werden. Ist ein Datensichtgerät vorhanden, so erscheint nur ein „j" am Bildschirm. Danach wird jedes Zeichen, das auf der Tastatur eingegeben wird, auch wieder auf den Bildschirm ausgegeben. Zu beachten ist beim Anschluß des Datensichtgeräts auch hier wieder, daß der Ausgang unseres Serien-Interface mit dem entsprechenden Daten-Eingang des Datensichtgerätes verbunden wird und umgekehrt. Ist man sich dabei nicht ganz sicher, so kann eine Messung mit einem Oszilloskop weiterhelfen. Dabei muß im Ruhezustand am Datenausgang des Sichtgerätes ein −12 V-Pegel liegen und bei Betätigung einer Taste ein kurzes Datenmuster erscheinen. Wird der so gefundene Ausgang mit dem entsprechenden Eingang des Datensichtgerätes verbunden, muß beim Drücken einer Taste auf dem Bildschirm ein Zeichen erscheinen. Geschieht dies nicht, so kann es sein, daß das Datensichtgerät einen DSR-Eingang besitzt. Dieser muß dann durch eine Brücke mit dem RTS-Ausgang desselben Datensichtgerätes verbunden werden. Funktioniert dieser Test, so kann der Eingang CTS des Datensichtgerätes mit unserem Ausgang RTS verbunden werden und umgekehrt unser Eingang CTS mit dessen Ausgang RTS.

7.4 Der Floppy-Anschluß

Hier folgt die Floppy-Steuer-Karte für den NDR-Klein-Computer. Jeder, der schon eine der Systemkarten erfolgreich aufgebaut hat, kann auch diese Karte nachbauen. Damit besitzt er dann die Schlüsselkarte, die seinen Computer zu einem professionellen Gerät macht. Übrigens kann die

Schaltung auch am mc-CP/M-Computer arbeiten, allerdings nur mit einem neuen EPROM und entweder über eine Adapterkarte oder über eine eigene Layout-Version.

mc hat schon in Heft 7/1984 über die Technik berichtet, mit der Daten auf Disketten geschrieben und auch wieder davon gelesen werden können. Das Thema ist deshalb nicht leicht zu behandeln, weil es dabei die verschiedensten Formate und Verfahren gibt. Unser Format wird nun, anders als damals beim mc-CP/M-Computer, 80-Spur-Laufwerke in 5¼-Zoll-Technik ansprechen und mit der MFM-Technik arbeiten. Weshalb diese Umstellung? Die technische Entwicklung ist seit September 1982, da erschien der mc-CP/M-Computer, so weit fortgeschritten, daß das damals verwendete Format nicht mehr zeitgemäß ist. Wir haben uns bemüht, jetzt einen zukunftssicheren Standard festzulegen. Da bei dieser Festlegung die Väter aller mc-Computer mitbestimmt haben, die für ihre Kinder nur das Beste akzeptieren, sind wir sicher, das Richtige gefunden zu haben, das jahrelang Bestand haben wird. Das mc-Format wird von allen mc-XXXXX-Computern, die nicht den 6502 als CPU besitzen, physikalisch gelesen werden können. Wir hoffen damit, daß mc-Leser bald problemlos zumindest Daten- und Text-Dateien austauschen können. Und daß Sie erkennen, daß mc-Systeme nicht ohne Überlegung entworfen werden.

FLO2 und seine Chips

Auf jeder Floppy-Karte sitzt mit dem Controller-Chip ein Baustein, der selbst ein sehr intelligenter Prozessor ist. Mit einigen wenigen Befehlen kann man komplexe Suchaktionen und vollständige Datenübertragungen auslösen, die vom Controller selbständig ausgeführt werden. Das hier verwendete IC 1797 (Lieferant zum Beispiel Siemens) besitzt unter anderem den Befehl Lies Spur, mit dem eine vollständige Diskettenspur komplett gelesen werden kann. Der Controller-Chip erledigt also die logischen und verwaltungstechnischen Aufgaben auf der FLO2-Platine. Er findet die richtige Spur, er meldet die Ergebnisse eines Suchprozesses an die CPU und er liefert auch die Daten dorthin ab. Dabei wird die Interrupt-Technik benutzt, um der CPU zu signalisieren, daß ein zu bearbeitendes Ergebnis vorliegt. Der zweite wichtige Baustein auf der Platine ist der Datenseparator-Baustein 9229, der die Schreib- und Lesesignale vom und zum 1797 so aufbereitet, daß sich die Elektronik des Laufwerkes und der Prozessor 1797 richtig verstehen. Im Prinzip geschieht dabei Folgendes (in mc 1984, Heft 7 ist das genau geschildert): Erstens wird das vom Floppy-Controller schon korrekt ausgegebene Schreibsignal zur Aufzeichnung in manchen Fällen noch etwas elektronisch zurechtgebogen, damit es später fehlerfrei wieder gelesen werden kann. Diese Vorbehandlung nennt man Präkompensation. Sie ist bei 8-Zoll-Laufwerken für die inneren Spuren notwendig (bei anderen Laufwerken meist überflüssig); zweitens nimmt der Baustein 9229 beim Lesen die Trennung zwischen den Synchronsignalen und den eigentlichen Datensignalen vor, die ja bei der Aufzeichnung zusammengemischt wurden. Aus diesem Grund heißt der Baustein 9229 auch Datenseparator. Daß er dabei Toleranzen ausregelt, die durch ungleichmäßigen Lauf der Magnetscheibe entstehen, sei hier nicht verschwiegen.

Weil der Separator-Baustein intern so raffiniert aufgebaut ist, kann man die Platine jetzt ohne Abgleich-Probleme aufbauen. Ohne weiteres kann man jetzt bei 8-Zoll-Laufwerken Double Density fahren. Man kann also jetzt jeden Laufwerktyp, sei es 3-Zoll, 3½-Zoll, 5¼-Zoll oder 8-Zoll, mit einfacher oder doppelter Dichte bedienen. Ebenso können Laufwerke mit nur einem Schreibkopf, Doppelkopf-Laufwerke und Laufwerke mit 80 Spuren verwendet werden. Auch gemischter Betrieb ist möglich.

Die Signale für FLO2

Über den bidirektionalen Bustreiber (IC 10, 74LS245) ist der Datenbus des Computers mit dem Controller-Baustein verbunden *(Abb. 7.4.1)*. Ein Widerstandsnetz mit acht Widerständen von je

3,3 kΩ sorgt dafür, daß keine Störungen auf der Versorgungsleitung auftreten, wenn der Buszustand am Controller von Tri State auf Aktiv wechselt. Solche Störungen können von 74LS245-ICs neuerer Generation verursacht werden, da diese bei einem Wechsel von Offen nach Aktiv schlagartig an den B-Eingängen viel Strom ziehen. Durch die Pull-up-Widerstände wird verhindert, daß die Spannung an den B-Eingängen dabei unter 2,5 V absinkt, was Signalverfälschungen verhindert.

Am Vergleicher (IC13, 74LS85) kann die Adresse der Baugruppe mit Brücken eingestellt werden. Sie lautet 0C0h, wenn die Brücken gegenüber von A4 und A5 auf Masse geschaltet werden und die Brücken gegenüber von A7 und A6 offenbleiben. Auf dem Bestückungsplan ist die Lage der Brücken eingezeichnet. Das IC gibt jetzt bei OUT genau dann 1-Pegel aus, wenn auf den Adreßleitungen die Signalkombination A4 = 0, A5 = 0, A6 = 1 und A7 = 1 auftritt. Von dem Decoder (IC14, 74LS138) werden zwei Adreßbereiche angesprochen. Von 0C0h bis 0C3h wird der Floppy-Controllerbaustein (IC 4) aktiviert und von 0C4h bis 0C7h werden beim Schreiben das IC12 und beim Lesen das IC11 angewählt. Dabei ist jedesmal aber nur die Adresse 0C4h gemeint, denn die anderen Adressen sind einfach nur nicht eindeutig decodiert und wählen ebenfalls die genannten Bausteine an.

Abb. 7.4.2 zeigt die Lötseite, *Abb. 7.4.3* die Bestückungsseite, *Abb. 7.4.4* den Bestückungsplan und *Tabelle 7.4.1* die Stückliste.

Das IC12 (74LS273) speichert die Laufwerksnummer und gibt sie an den Stecker zum Laufwerk. Außerdem werden Laufwerktyp (Mini oder Maxi) sowie Schreib- und Lesedichte (single, double) und Seitenauswahl-Signal vom Prozessor dort erwartet.

Sowohl die Adressierung, damit die Karte sich angesprochen fühlt, als auch die Anwahl und Einstellung der einzelnen Laufwerke muß also die CPU durchführen.

Abb. 7.4.5 zeigt die Bedeutung der Bits in IC12. Die Bits 0 und 3 dienen zum Einstellen des Laufwerks. Wenn ein Laufwerk angesprochen werden soll, so wird einfach das entsprechende Bit auf 1 gesetzt. Dabei ist die Laufwerkscodierung also nicht dual durchgeführt, sondern durch Anwahl von genau einer von 4 Leitungen. Wenn man über Laufwerke verfügt, die intern eine duale Anwahl verkraften können, dann könnte man bis zu 16 Laufwerke anschließen. Solche Laufwerke sind selten.

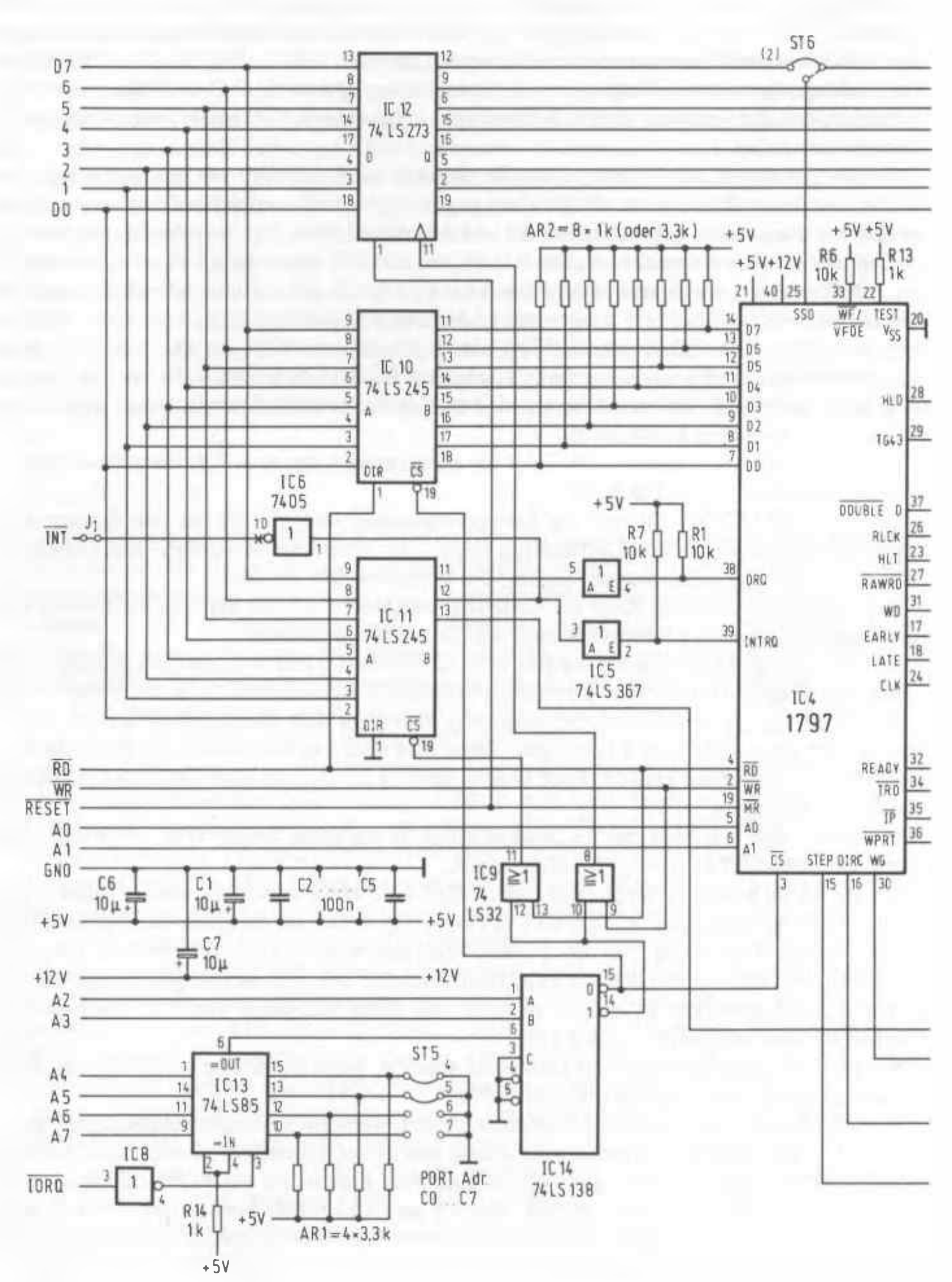
Mit Bit 4 wird bestimmt, ob die Aufzeichnung in einfacher Dichte (FM, Bit 4 = 1) oder doppelter Dichte (MFM, Bit 4 = 0) erfolgen soll.

Mit Bit 5 wird bestimmt, ob 5¼-Zoll- oder 8-Zoll-Laufwerke verwendet werden sollen. Wird das Bit 5 auf 1 gesetzt, so ist der Takt für den Floppy-Controller auf die Hälfte herabgesetzt. Das ist die richtige Einstellung für die 5¼-Zoll-Mini-Laufwerke. Auch die meisten kleineren Laufwerke, die Mikrolaufwerke, kommen damit zurecht. Die Aufzeichnungsfrequenz bei den großen 8-Zoll-Laufwerken ist doppelt so groß, was durch Nullsetzen von Bit 5 auch auf der Controller-Platine eingestellt werden kann.

Mit Bit 6 kann man den Motor der Laufwerke schalten, wenn die Brücke bei ST3 so eingebaut ist, wie sie im Bestückungsplan eingezeichnet ist.

Wenn Bit 6 auf 0 steht, so ist der Motor eingeschaltet, wenn auf 1, so ist er ausgeschaltet. Wenn man das Bit zum Schalten verwenden will, so muß man darauf achten, daß entweder das Laufwerk einen READY-Ausgang besitzt oder daß man nach dem Einschalten per Warteschleife so lange wartet, bis der Motor seine Nenndrehzahl erreicht hat. Andernfalls können insbesondere beim Schreiben Fehler auftreten, die defekte Sektoren verursachen. Beim Lesen ist das anders. Da wird ein Sektor vom Controller automatisch immer wieder gelesen, wenn sich bei der Fehlerüberprüfung herausstellt, daß er von der Diskette fehlerhaft abgetastet wurde.

Mit Bit 7 kann man die Floppy-Seite auswählen, wenn man Laufwerke mit zwei Köpfen besitzt. Dazu muß die Brücke St6, wie im Schaltbild eingezeichnet, verdrahtet sein. Im Layout ist die Brücke bereits fest eingebaut, daher muß man normalerweise keine Veränderung an der


D7
6
5
4
3
2
1
D0
IC 12
74 LS 273
IC 10
74 LS 245
IC 11
74 LS 245
IC6
7405
INT
J1
IC5
74LS 367
AR2 = 8 × 1k (oder 3,3k)
+5V
+12V
R6
10k
R13
1k
R7
10k
R1
10k
ST6
IC4
1797
D7
D6
D5
D4
D3
D2
D1
D0
SSO
WF/VFOE
TEST
VSS
HLD
TG43
DOUBLE D
RLCK
HLT
RAWRD
WD
EARLY
LATE
CLK
DRQ
INTRQ
READY
TRO
IP
WPRT
RD
WR
MR
A0
A1
CS
STEP
DIRC
WG
RD
WR
RESET
A0
A1
GND
C6
10µ
C1
10µ
C2 ... C5
100n
C7
10µ
+5V
+12V
A2
A3
A4
A5
A6
A7
IORQ
IC9
74
LS32
IC14
74LS138
IC13
74LS85
IC8
ST5
PORT Adr.
C0 ... C7
R14
1k
AR1 = 4 × 3,3k

zu Abb. 7.4.1

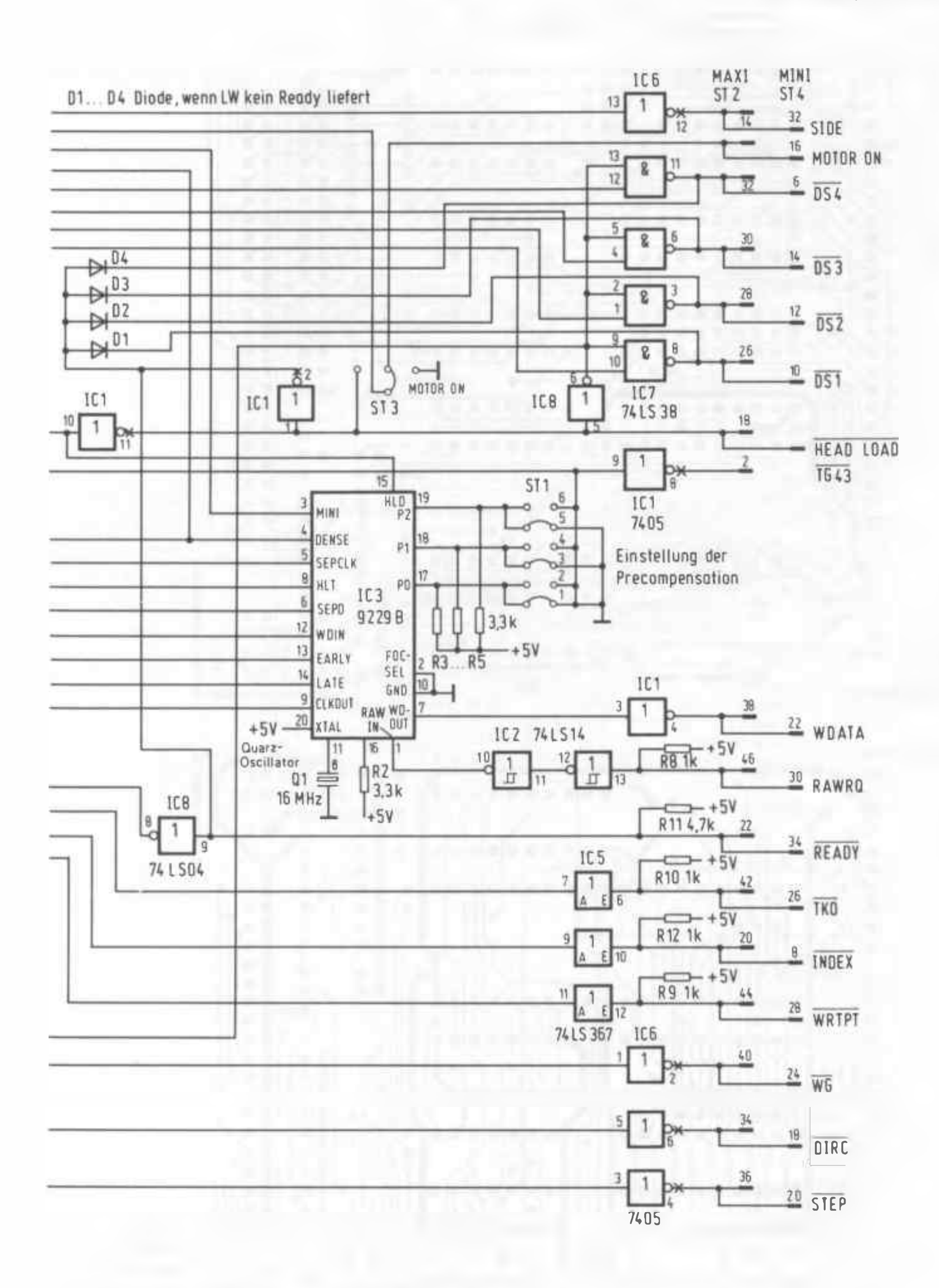


Abb. 7.4.1 Der Schaltplan der FLO2-Baugruppe

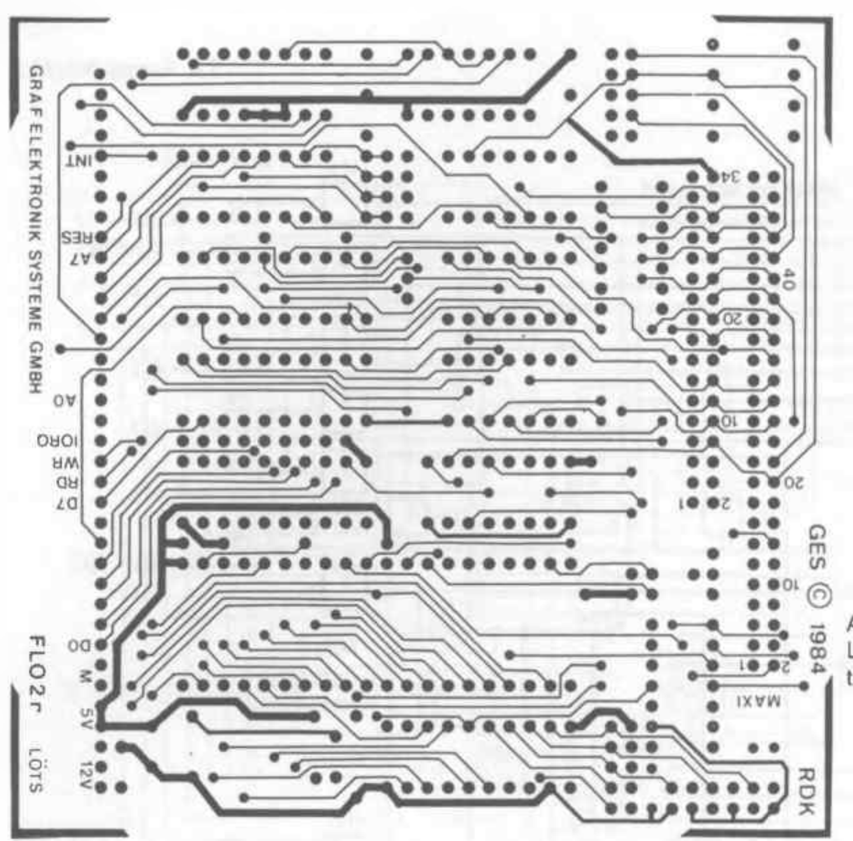


Abb. 7.4.2 Die Lötseite der Leiterplatte FLO2

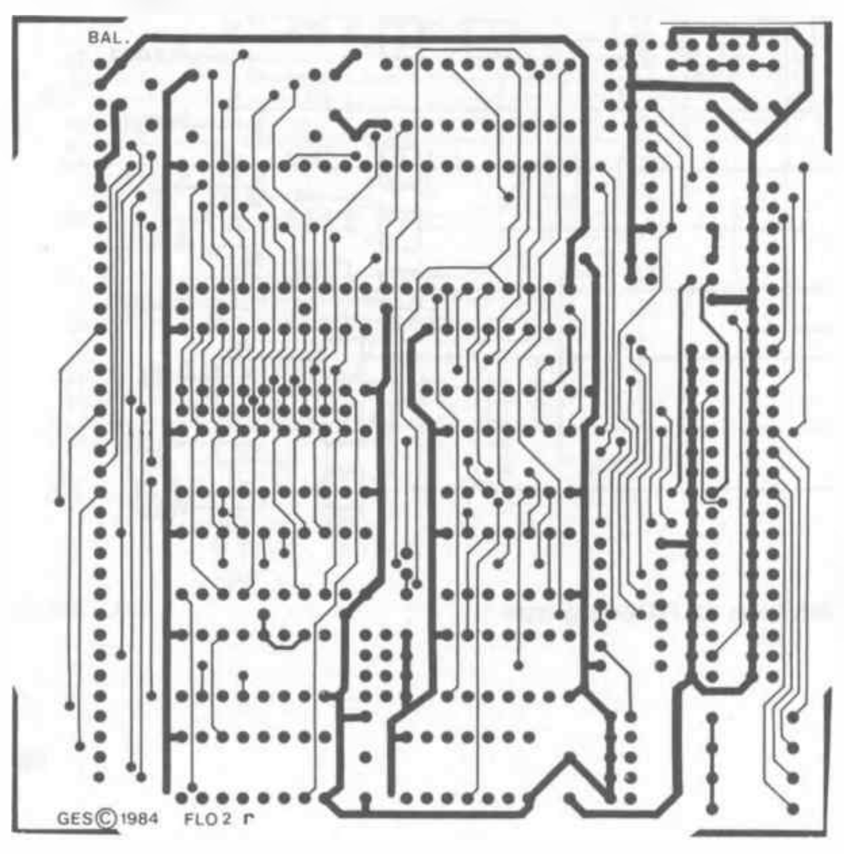


Abb. 7.4.3 Die Bestückungsseite der Leiterplatte FLO2

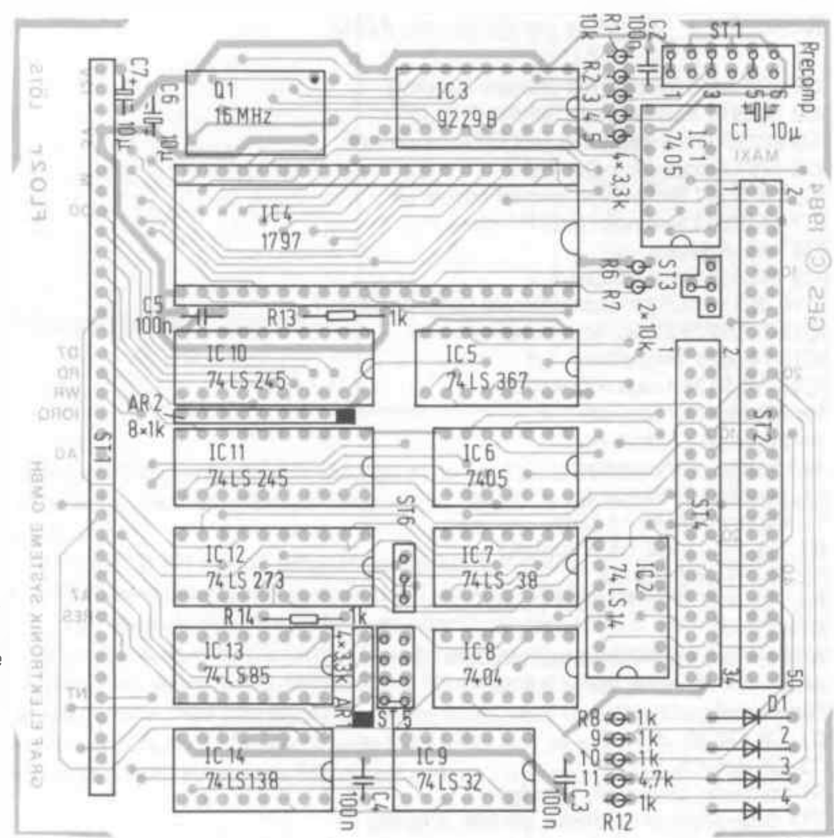


Abb. 7.4.4 Bestückungsplan der Baugruppe FLO2

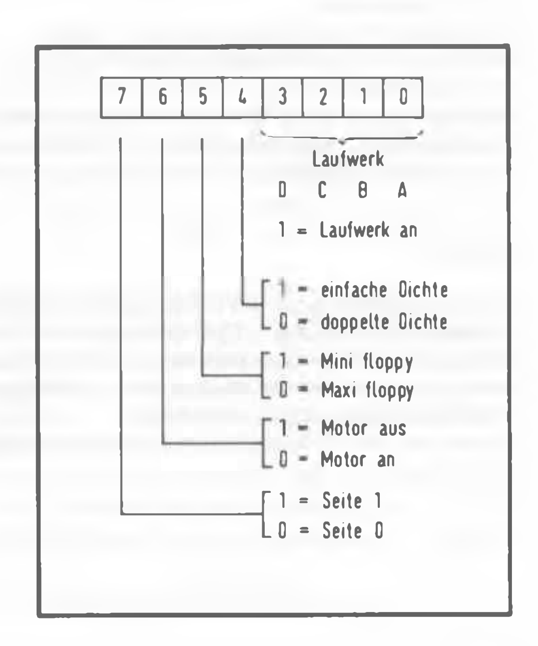


Abb. 7.4.5 Die Bedeutung der Bits in Port C4 beim Schreiben

Tabelle 7.4.1 Stückliste zur Baugruppe FLO2

IC1, IC6 7405 Inverter mit offenem Kollektor
IC2 74LS14 Invertierende Schmitt-Trigger
IC3 9229B Datenseparator und Präkompensator
IC4 1797 Floppy-Disk-Controller
IC5 74LS367 Nicht-Invertierende Treiber
IC7 74LS38 Leistungs-Nand-Glieder
IC8 7404 Inverter
IC9 74LS32 Oder-Glieder
IC10, IC11 74LS245 bidirektionale Datenbustreiber
IC12 74LS273 Zwischenspeicher
IC13 74LS85 Vergleicher
IC14 74LS138 1-aus-8 Dekoder
1 × 40polige IC-Fassung
3× 20polige IC-Fassung
3x 16polige IC-Fassung
6× 14polige IC-Fassung
1 × Quarzoszillator 16 MHz
R1, R6, R7 10 kΩ 1/8 W (unkrit.)
R2, R3, R4, R5 3.3 kΩ
R8, R9, R10, R12, R13, R14 1 kΩ
R11 4.7 kΩ
AR1 4 × 3.3 kΩ Widerstandsnetzwerk
AR2 8 × 1 kΩ (oder früher 3.3 kΩ)
Widerstandsnetzwerk
C1, C6, C7 10 µF 16 V
C2, C3, C4, C5 100 nF
ST Stiftleiste, gewinkelt, 36polig
ST1 doppelreihige Stiftleiste, gerade, 2*6polig
ST2 doppelreihige Stiftleiste, gewinkelt, 2*25polig
ST3 5 Einzelstifte
ST4 doppelreihige Stiftleiste, gerade, 2*17polig
ST5 doppelreihige Stiftleiste, gerade, 2*4polig
1 × Leiterplatte FLO2
1 × Verbindungskabel Flachbandleitung, passend zum Disketten-Laufwerk
1 × Spannungsversorgungs-Kabel für Diskettenlaufwerk
1 × Diskettenlaufwerk (z. B. TEAC, 80-Spur, doppelseitig).

Kenndaten:

Spannungsversorgung: + 5 V, Stromaufnahme 360 mA
Spannungsversorgung: + 12 V, Stromaufnahme 20 mA
Die Laufwerke benötigen auch eine zusätzliche Spannungsversorgung. Je nach Laufwerk sind + 5 V und + 12 V bei den meisten Mini- und Mikrolaufwerken oder + 5 V und + 24 V bei den meisten Maxi-Laufwerken erforderlich.
Dazu muß man im Datenblatt der Laufwerke nachsehen.

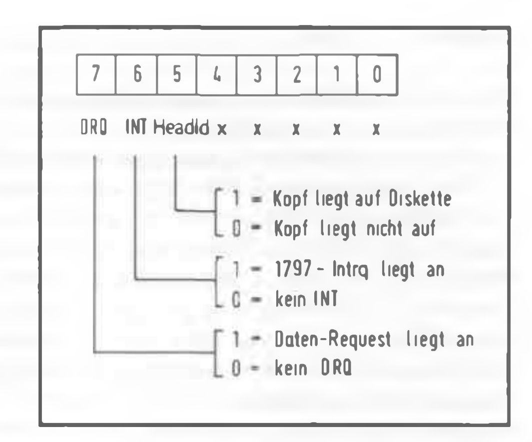


Abb. 7.4.6 Die Bedeutung der Bits in Port C4 beim Lesen

Leiterplatte vornehmen. Wenn man das Bit 7 auf 1 setzt, so wird Seite 1, also die Rückseite der Floppy-Scheibe (das ist die mit Beschriftung!) ausgewählt. Seite 0 ist angewählt, wenn man das Bit 7 auf den Wert 0 setzt.

Der Floppy-Controller FD1797 besitzt selbst ebenfalls einen Ausgang für die Seitenauswahl, den Ausgang SS0. Die Verwendung dieses Ausgangs hat jedoch große Nachteile. Wenn man ihn verwendet, so kann man nämlich nicht verhindern, daß der Controller anhand des Datenrecords automatisch kontrolliert, ob auch wirklich Seite 1 angesprochen wird. Es gibt aber Disketten mit Programmen im Handel, bei welchen das Seitenbit der Sektorkennzeichnung bei der Formatierung nicht auf 1 gesetzt wurde. Will man solche Disketten lesen, muß man dem Controller einen Befehl geben können, nach dem er glaubt auf Seite 0 zu lesen und zu prüfen, während Seite 1 angewählt ist. Das gelingt nur, wenn man SS0 selbst erzeugt, wie zum Beispiel auf FLO2.

So wird gelesen

Abb. 7.4.6 zeigt die Bedeutung der Bits unter Adresse 0C4h beim Lesen. Bit 7 ist das DRQ-Bit des Controllers. Leider kann man es nicht ohne Nachteile vom Controller direkt abnehmen. Daher muß es über das IC 11 geführt werden. Wenn dieses Bit auf 1 liegt, so will der Controller ein Byte holen oder hat ein Byte bereitgestellt. DRQ wurde auf Bit 7 des Datenbusses gelegt, damit man eine schnelle Abfrage des Bits (durch einen RLCA-Befehl beim Z80 oder ROL.B-Befehl beim 68000/8) erreichen kann. Das Bit landet nämlich schnell im Carry-Flag. Man kann danach durch einen Sprung „JP C," bzw. „JP NC," oder „BCS" bzw. „BCC" rasch eine Verzweigung durchführen.

Genauso verhält es sich mit dem Bit 6, dort liegt der INTRQ-Ausgang des Controllers. Das Bit wird auf 1 gesetzt, wenn der Controller eine Unterbrechung des Hauptprozessors erreichen will, weil er etwas mitzuteilen hat. Nun kann man zum einen einen echten Interrupt auslösen, wie es bei der Z80-CPU vorgesehen ist, oder das Bit abfragen, wie es bei unseren 68000/8-Routinen getan wird. Das Bit landet nach einem RLCA-Befehl (ROL.B) nämlich im Vorzeichenbit. Mit „JP M" (JP P, BMI, BPL) kann dann eine Entscheidung getroffen werden.

Bit 5 enthält die Headload-Information: Wenn der Kopf auf der Diskette aufliegt, ist das Bit auf 1 gesetzt. Damit kann man vor einem Schreib- und Lesebefehl abfragen, ob der Kopf noch auf der Diskette aufliegt. Wenn ja, braucht man ihn nicht zu laden und kann dadurch 15 ms Kopfladezeit sparen.

Die restlichen Bits sind unbelegt und können später einmal Erweiterungen dienen.

Das IC FDC-9229B

Das IC FDC-9229B ist ein universeller Baustein, der den Aufbau von Floppy-Controller-Schaltungen stark vereinfacht.

Das IC beinhaltet neben dem Datenseparator auch eine Präkompensationsschaltung.

Der Datenseparator ist sehr ausgeklügelt. So ist zum Beispiel kein Abgleich erforderlich, was den Nachbau der FLO2 erheblich vereinfacht. Der Präkompensator besitzt drei Eingänge P0, P1 und P2, mit welchen man seine Zeiten flexibel einstellen kann. In der Schaltung wird das mit den Brücken bei ST1 getan. Dabei ist wichtig, daß manche Floppy-Laufwerke eine Präkompensation erst ab Spur 43 benötigen. Es gibt daher drei Möglichkeiten für jeden der Eingänge P0 bis P2. Einmal liegt der Eingang über einen Widerstand an + 5 V. Dann ist die Präkompensation dauerhaft eingestellt. Zum zweiten kann der Eingang auf 0 V geschaltet werden. Dann ist das betreffende Bit unwirksam. Wird der Eingang zum dritten auf den Ausgang TG43 des Floppy-Controllers geschaltet, dann ist die Präkompensation erst dann wirksam, wenn eine Spur größer gleich 43 angefahren wird.

Für moderne 5¼-Zoll-Laufwerke genügt es, alle Eingänge auf 0 zu schalten, also keine Präkompensation zu verwenden. In den technischen Handbüchern der Laufwerke steht ein Hinweis, falls eine Präkompensation erforderlich ist.

Tabelle 7.4.2 enthält die möglichen Präkompensationszeiten. Die Zeiten sind unterschiedlich, je nachdem ob die Einstellung Mini oder Maxi vorliegt.

Abb. 7.4.7 zeigt die Einstellung für ein Maxi-Laufwerk mit mehr als 40 Spuren, das eine Präkompensation von 62,5 ns benötigt, wenn eine Spur größer gleich 43 gewählt wird. Man kann auch Präkompensationszeiten einstellen, die auf allen Spuren gleich sind oder ab einer bestimmten Spur einen höheren Wert annehmen. Aber, wie schon gesagt, die meisten Laufwerke benötigen keine Präkompensation. Das gilt insbesondere für Minilaufwerke mit 40 Spuren. Präkompensation ist übrigens nur bei doppelter Aufzeichnungsdichte überhaupt relevant.

Tabelle 7.4.2 Die Präkompensationszeiten des 9229

Mini = 0 (also Maxi)

P2	P1	P0	
0	0	0	0 ns
0	0	1	62,5 ns
0	1	0	125 ns
0	1	1	187,5 ns
1	0	0	250 ns
1	0	1	250 ns
1	1	0	312,5 ns
1	1	1	312,5 ns

Mini = 1 (also Mini)

P2	P1	P0	
0	0	0	0 ns
0	0	1	125 ns
0	1	0	150 ns
0	1	1	375 ns
1	0	0	500 ns
1	0	1	500 ns
1	1	0	625 ns
1	1	1	625 ns

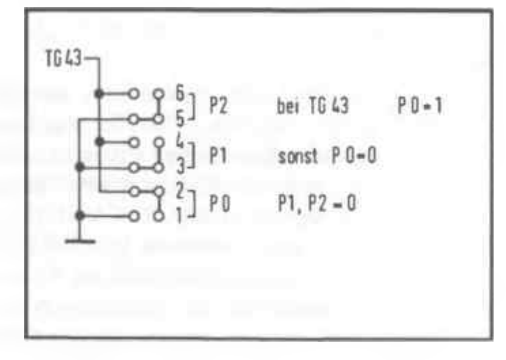


Abb. 7.4.7 Eine Einstellung der Präkompensation als Beispiel

Bei der Präkompensation werden nahe beieinander stehende Datenbits für die Aufzeichnung noch näher zusammengedrängt. Denn bei einer Wiedergabe erscheinen sie gegeneinander verzögert, da sie vom Kopf beim Lesen „auseinandergerückt“ werden. Die Präkompensationszeit gibt an, um wieviel die Bits zusammengedrängt werden. Wählt man diese Zeit zu groß, so gibt es Datenfehler, denn dann kann der Controller die Daten nicht mehr lesen. Also für die ersten Versuche die Brücken 1, 3 und 5 bei ST1 einsetzen.

Die Steuerleitungen

Eine wichtige Funktion erfüllt das Ready-Signal, das von dem angesprochenen Laufwerk kommt. Bei allen Maxi-Laufwerken ist es vorhanden und liegt auf Pin 22 der 50poligen Steckerleiste. Bei Minilaufwerken ist es oft nicht vorhanden. Bei Laufwerken, die keinen Kopflademagnet besitzen und die den Motor daher abschalten müssen, wenn sie nicht angesprochen werden, ist immer ein Ready-Signal vorhanden. Wenn ein Ready-Ausgang vorhanden ist, so liegt er meist auf dem Stift 34 der 34poligen Steckerleiste. Er muß dann in unserer Schaltung per Hand auf den gemeinsamen Ready-Eingang der Dioden (Anode) D1 bis D4 verdrahtet werden. Die Dioden selbst dürfen dann nicht eingesetzt werden. Wer den Schaltplan genau analysiert, stellt fest, daß sowohl $\overline{\text{READY}}$ als auch IC1 dieselbe Datenleitung bewegen. Das geht ohne Kurzschluß, weil beide Signale von Open-Collector-Ausgängen herrühren.

Die Dioden werden für die Laufwerke benötigt, die keinen Ready-Ausgang besitzen. Wenn man gemischt arbeitet, dürfen die Dioden nur für solche Laufwerke eingesetzt werden, die keinen Ready-Ausgang besitzen (auch bei gemischtem Mini- und Maxilaufwerksbetrieb). Die Diode D1 ist dabei für das Laufwerk A (Drive select 1), die Diode D2 für das Laufwerk B usw. zuständig. Man kann insgesamt vier Laufwerke anschließen, wenn man die „eins aus vier“ Codierung verwendet. Dabei kann jedes Laufwerk zusätzlich doppelseitig sein. *Abb. 7.4.8* zeigt Beispiele für die Diodenbestückung. Bei binärer Codierung sind im Prinzip 15 Laufwerke möglich. Man muß dabei mit der Ready-Leitung aufpassen. Da dies aber nur ganz seltene Laufwerke betrifft, sei hier nicht weiter drauf eingegangen. Übrigens liegt Ready an der Stiftleiste auf 0, wenn das Laufwerk bereit ist.

Die anderen Steuer- und Meldesignale sind nicht so aufregend: Mit Side wird die Laufwerksseite bestimmt. Side liegt auf 0, wenn die Seite 1 ausgewählt wird, bei Seite 0 liegt es auf 1.

Motor On wird meist nur für Minilaufwerke verwendet. Es liegt auf 0, wenn der Motor an sein soll.

Maxi A-D	Mini A-D	gemischt A,B Maxi C,D Mini
o offen o D1	o —▷|— o D1	o o
o offen o D2	o —▷|— o D2	o o
o offen o D3	o —▷|— o D3	o —▷|— o
o offen o D4	o —▷|— o D4	o —▷|— o

Abb. 7.4.8 So werden die Dioden eingestellt

$\overline{\text{DS1}}$ bis $\overline{\text{DS4}}$ liegen auf 0, wenn das entsprechende Laufwerk angesprochen wird. Achtung, bei manchen Minilaufwerken gibt es kein $\overline{\text{DS4}}$. Mit Head Load wird der Laufwerkskopf an die Diskettenoberfläche gebracht, wenn das Signal auf 0 liegt. Es ist bei uns mit dem Selekt-Signal verkoppelt.

$\overline{\text{TG43}}$ liegt auf 0, wenn eine Spur größer gleich 43 angesprochen wird. Dieses Signal wird nur von älteren Maxi-Laufwerken verwendet, um den Schreibstrom zu reduzieren. Manchmal liegt der Anschluß bei dem Laufwerk auch an Stift 8 und nicht an 2.

WDATA sind Schreibdaten. RAWRQ sind die Lesedaten. $\overline{\text{TK0}}$ liegt auf 0, wenn der Kopf auf Spur 0 liegt. Es gibt ein paar Laufwerke, bei welchen die Positionierung auf Spur 0 nicht funktioniert. Dort kann der Kopf auch noch auf Spur 1 fahren, was die Spur-0-Meldeschaltung durcheinanderbringen kann. Meist läßt sich das durch eine mechanische Neueinstellung beheben.

$\overline{\text{WRTPRT}}$ ist der Schreibschutzausgang. Damit kann das Laufwerk signalisieren, daß ein Schreibschutz gesetzt ist.

$\overline{\text{WG}}$ ist ein Freigabesignal für den Schreibausgang. $\overline{\text{DIRC}}$ und $\overline{\text{STEP}}$ dienen der Ansteuerung des Schrittmotors zur Spureinstellung.

Der Floppy-Controller

Der Floppy-Controller wird über die Adressen 0C0h bis 0C3h angesprochen. *Tabelle 7.4.3* zeigt die Belegung. Die Adresse 0C0h spricht beim Schreiben das Befehlsregister an. Man kann dort Befehle an den Controller übergeben. Beim Lesen erfährt man den Status, so zum Beispiel, ob ein Lesefehler aufgetreten ist oder nicht.

	Lesen	Schreiben	
C0	Status	Befehl	
C1	Spur-Register	Spur-Register	
C2	Sektor-Register	Sektor-Register	1797
C3	Daten-Register	Daten-Register	
C4	Zusatz-Info	Auswahl	Zusatz

Tabelle 7.4.3 Die Bedeutung der Register des Floppy-Controllers

Das Register 0C1h ist das Spur-Register, dort steht die aktuelle Spurnummer, die von 0 bis maximal 255 reichen darf. Bei 8-Zoll-Laufwerken ist die größte Spurenzahl 76 und bei Minilaufwerken 34, 39 oder 79.

0C2h ist das Sektorregister, dort steht die aktuelle Sektornummer, auf die zugegriffen wird. Im Register 0C3h werden Parameter und Daten vom Controller an den Computer oder vom Computer an den Controller gegeben.

0C4h gehört, wie schon bekannt, nicht zum Floppy-Controller. Die *Tabelle 7.4.4* zeigt eine Liste der Befehle. Man unterscheidet vier Typen. Die erste Gruppe (I) sind Befehle für die Kopfpositionierung, die Gruppe II sind Befehle zum Lesen und Schreiben von Sektoren, die Gruppe III sind Befehle für die Kontrolle oder zum Formatieren. In der Gruppe IV sitzt nur ein Befehl für die Interruptsteuerung.

Tabelle 7.4.4 Die Befehle des 1797

Typ	Befehl	Befehl	7	6	5	4	3	2	1	0
I	Restore	Auf Spur 0	0	0	0	0	h	V	R0	R1
I	Seek	Spur suchen	0	0	0	1	h	V	R0	R1
I	Step	schreiten	0	0	1	u	h	V	R0	R1
I	Step In	schreite nach Innen	0	1	0	u	h	V	R0	R1
I	Step Out	schreite nach Außen	0	1	1	u	h	V	R0	R1
II	Read Sektor	Lies Sektor	1	0	0	m	F_2	E	F_1	0
II	Write Sektor	Schreibe Sektor	1	0	1	m	F_2	E	F_1	a_0
III	Read Address	Lies Adresse	1	1	0	0	0	E	F_1	0
III	Read Track	Lies Spur	1	1	1	0	0	E	F_1	0
III	Write Track	Schreibe Spur	1	1	1	1	0	E	F_1	0
IV	Force Interrupt	Interrupt auslösen	1	1	0	1	I_3	I_2	I_1	I_0

Generell wird vom Controller immer nach Ausführung eines Befehls die Leitung INT aktiviert und damit das Bit 6 am Port 0C4h gesetzt sowie INT auf 0 gesetzt. Wenn der Interrupt der CPU durch einen EI-Befehl beim Z80 (oder entsprechendes beim 68000/8) vorher freigegeben war, wird der Prozessor also zur Reaktion gezwungen. Wenn die CPU dann das Status-Register des Controllers liest, so wird das Signal INTRQ wieder gelöscht, also Bit 6 auf 0 zurückgesetzt. Bei den Befehlen sind einige Optionen möglich. So kann man die Steprate bestimmen, mit der die Kopfpositionierung durchgeführt wird. *Tabelle 7.4.5* zeigt die Umrechnungstabelle. Dann gibt es noch einige Hilfsbits, deren Bedeutung *Tabelle 7.4.7* zeigt. Wenn man das Bit mit der Bezeichnung „h“ auf 1 setzt, so wird der Kopf zu Beginn des entsprechenden Befehls geladen, sonst erfolgt nur ein Update der internen Register. Bei den STEP-Befehlen muß zum Update zusätzlich das Bit U auf 1 gesetzt sein.

Wird das Bit V auf 1 gesetzt, so erfolgt ein Prüfvorgang. Der Kopf eines Records wird angelesen und der Inhalt mit der aktuellen Spurnummer und ggf. Seitennummer verglichen. Stimmen die Werte nicht überein, erfolgt eine Fehlermeldung.

R1	R0	Maxi	Mini	Monitor
0	0	3 ms	6 ms	0
0	1	6 ms	12 ms	1
1	0	10 ms	20 ms	2
1	1	15 ms	30 ms	3

Tabelle 7.4.5 Die Stepraten

Tabelle 7.4.6 Die Bedeutung der speziellen Bits aus Tabelle 7.4.4

h = 1	Kopf bei Start laden
h = 0	Kopf bei Start heben
v = 1	Spur prüfen durch Anlesen
v = 0	keine Prüfung

u = 1	Spur-Register auf Stand bringen
u = 0	Spur-Register belassen
m = 0	einen Record bearbeiten
m = 1	mehrere Records bearbeiten
$d_0 = 0$	Data-Mark FB
$d_0 = 1$	Deleted Data-Mark F8
$F_2 = 1$	Sektorlänge = Std. (IBM) 128, 256, 512, 1024
$F_2 = 0$	spezielle Längen = 256, 512, 1024, 128
$F_1 = 0$	SSO auf 0 (Seite 0 prüfen)
$F_1 = 1$	SSO auf 1 (Seite 1 prüfen)
$I_0 = 1$	Nicht Ready → Ready gibt INT
$I_1 = 1$	Ready → Nicht Ready gibt INT
$I_2 = 1$	Index Puls
$I_3 = 1$	sofort INT auslösen
$I_3–I_0 = 0$	Stop ohne Interrupt

Beim Befehl „Spur suchen" wird die einzustellende Spur im Port-Register 0C3h übergeben. Step schreitet einen Schritt in die Richtung in die zuletzt geschritten wurde, Step In schreitet einen Schritt in Richtung Spur 76 (80 usw.). Step Out schreitet einen Schritt in Richtung Spur 0. Beim Schreiben oder Lesen wird zusätzlich die Sektorinformation im Register 0C2h ausgewertet. Dieser Sektor wird angewählt. Normalerweise fängt man bei Sektoren mit 1 an zu zählen, bei Spuren mit 0. Die Daten werden im Register 0C3h ausgetauscht. Beim Schreiben kündigt das DRQ-Signal die Anforderung eines neuen Wertes an, beim Lesen zeigt DRQ an, daß ein Byte vorliegt. Das DRQ-Bit wird jeweils gelöscht, wenn man ein Byte geliefert oder geholt hat. Der Controller gibt eine Fehlermeldung aus, wenn man dabei so spät reagiert, daß er nicht mehr fortlaufend mit dem Takt der Aufzeichnung auf der Floppy arbeiten kann.

Die Belegung des Statusregisters zeigt *Tabelle 7.4.7*. Sie ist abhängig vom ausgeführten Befehl und unterscheidet Typ-I- und Typ-II/III-Befehle. Mit dem Befehl „Read Adress" kann man den Startkopf eines Sektors lesen, der Auskunft über aktuelle Spur, Seite, Sektor und Sektorlänge gibt. Die Information wird in der angegebenen Reihenfolge nacheinander im Register 0C3h, also dem Datenregister mit einer DRQ-Anforderung übergeben.

Mit dem Befehl „Lies Spur" wird eine komplette Spur, beginnend beim Index Loch bis zum erneuten auftreten des Indexes eingelesen. Dabei werden nicht nur die Daten übertragen, sondern

Tabelle 7.4.7 Die Bedeutung der Bits im Statusregister

	7	6	5	4	3	2	1	0
Befehls-typ I	1 = Nicht Ready	1 = Schreib-schutz	1 = Kopf geladen	1 = Such-fehler	1 = CRC-Fehler	1 = Spur 0	1 = Index	1 = Busy
Befehls-typ II, III	1 = Nicht Ready Der Be-fehl wird nicht aus-gef. bis Ready 0	1 = Schreib-schutz	1 = Typ-Fehler 1 = Dele-ted Data Mark oder Schreib-fehler	1 = Record nicht gefunden	1 = CRC-Fehler	1 = Daten-verlust	1 = DRQ	1 = Busy

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF (or 00)[1]
6	00
1	FC (Index Mark)
26	FF (or 00)
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00
1	F7 (2 CRC's written)
11	FF (or 00)
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	FF (or 00)
247**	FF (or 00)

*Write bracketed field 26 times

**Continue writing until FD179X interrupts out. Approx. 247 bytes.

1-Optional '00' on 1795/7 only.

Abb. 7.4.9 IBM-Formatierung mit SD. Nach dem Befehl Schreibe Spur müssen dem Controller auf Anforderung diese Bytes der Reihe nach zur Verfügung gestellt werden

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F6
1	FC (Index Mark)
50*	4E
12	00
3	F5
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01
1	F7 (2 CRCs written)
22	4E
12	00
3	F5
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRCs written)
54	4E
598**	4E

* Write bracketed field 26 times

**Continue writing until FD179X interrupts out. Approx. 598 bytes.

Abb. 7.4.10 Bei MFM gehen genau doppelt soviele Bytes in eine Spur

auch alle Zusatzinformationen und Datenlücken, also überhaupt alles, was sich auf der Spur befindet. Die Daten sind immer korrekt, da sich der Controller bei Synchrontakten immer wieder neu auf den Floppy-Takt einregelt. Dieser Befehl ist weniger zum Lesen aktueller Daten als vielmehr zu Kontrollzwecken gedacht. Der Befehl „Write Track" schließlich dient zum Formatieren einer Diskette. Bestimmte Bytes werden dabei als Steuerinformation zur Ablage von CRC oder Synchronisationsbits interpretiert. *Tabelle 7.4.8* zeigt die Bedeutung des Bytes. *Abb. 7.4.9* zeigt als Beispiel die Formatierung nach IBM mit einfacher Dichte auf 8 Zoll. *Abb. 7.4.10* zeigt ein Beispiel für die Formatierung mit doppelter Dichte, auch auf 8 Zoll.

Tabelle 7.4.8 Die Formatier-Bytes

	einfache Dichte	doppelte Dichte
00–F4	Daten FM	Daten MFM
F5	–	A1* MFM, CRC löschen
F6	–	C2** MFM
F7	2 CRC-Bytes	2 CRC-Bytes
F8–FB	F8–FB mit CLK = C7, CRC löschen	F8–FB MFM
FC	FC mit CLK = D7	FC MFM
FD	FD mit CLK = FF	FD MFM
FE	FE mit CLK = C7, CRC löschen	FE MFM
FF	FF mit CLK = FF	FF MFM

* 4,5 Taktbit auslassen
** 3,4 Taktbit auslassen

Der eigentliche Datentransfer geschieht mit den Befehlen Lese Sektor und Schreibe Sektor, nachdem die gewünschte Spurnummer und die richtige Sektornummer eingestellt wird. Dann bietet der Controller die Bytes im Sektor im Abstand von 18 µs (große Floppy DD) oder 36 µs (Mini-Floppy) an seinen Datenleitungen an oder erwartet sie dort. Nun bleibt noch der Befehl „Force Interrupt". Damit kann man zum einen den Controller rücksetzen, zum anderen das Interruptverhalten einstellen, wie in Abb. 7.4.13, bei den Bits I0 bis I3 dargestellt.

Zum Aufbau und zum Test

Beim Aufbau beginne man zunächst mit dem Einsetzen der IC-Fassungen. Dann werden alle passiven Bauteile, wie Widerstände und Kondensatoren eingelötet und auch die Stiftleisten.

Als nächstes löte man den Quarzoszillator Q1 ein. Dabei ist auf die Orientierung zu achten. Der Quarzoszillator besitzt an einer seiner Gehäuseseiten einen Punkt. Dieser Punkt kennzeichnet Pin 1. Er muß mit dem Punkt auf dem Bestückungsplan übereinstimmen. Beim Quarzoszillator darf man nicht zu lange löten. Da der Baustein nur vier Anschlüsse besitzt, wäre die mechanische Stabilität nicht sehr hoch, wenn man den Oszillator mit einer Fassung montieren würde.

Nach dem Einbau des Oszillators kann man alle ICs einstecken. Der Floppy-Controller benötigt eine zusätzliche Versorgungsspannung von + 12 V, die über den Bus geführt wird. Bitte das Vorhandensein dieser Spannung nochmals kontrollieren, bevor die Baugruppe in Betrieb genommen wird. *Abb. 7.4.11* zeigt eine Zusammenfassung aller Brücken, wie sie auf die unterschiedlichen Stiftleisten (Ausnahme ST6) gesteckt werden müssen.

Für den Test gilt:

1. Betrieb zunächst ohne Floppy-Laufwerk. Zum Test wird entweder die SBC2-Baugruppe mit Grundprogramm allein oder die Vollausbau-CPU mit Speicher und Grundprogramm auf der Bank/Boot oder die mc-CP/M-CPU mit dem Monitorprogramm verwendet. Beim 68000/8 wird das Grundprogramm zum Test benötigt.

Nach dem Einschalten darf zunächst nichts weiter passieren, das Grundprogramm, bzw. der Monitor müssen sich melden.

Nun gebe man den Wert 55 auf IO-Port C1 aus, das geschieht mit dem Befehl „IO-Setzen". *Abb. 7.4.12*. Wenn man nun mit „IO-Lesen" den Inhalt des Ports wieder abfragt, so muß 55 wieder erscheinen, wie *Abb. 7.4.13* zeigt.

Nun führe man den Test noch mit dem Wert AA aus. Man hat damit geprüft, ob alle Datenleitungen o. k. sind und ob die Adressierung stimmt. Jetzt geht es an das Laufwerk. Das Laufwerk (Mini oder Maxi) wird über ein Flachbandkabel an die entsprechende Stiftleiste der FLO2-Baugruppe angesteckt. Dazu sollte man zuvor die Anleitung des Laufwerks studieren, um alle Einstellungen beim Laufwerk richtig setzen zu können. Das Laufwerk muß auf Laufwerksadresse 1, DS1, A, also als das erste eingestellt werden. Das Laufwerk benötigt eigene Versorgungsspannungen. Bei Minilaufwerken werden neben + 5 V meist noch + 12 V und bei Maxilaufwerken neben + 5 V noch + 24 V, und ggf. einige andere Spannungen verlangt.

Jetzt kann der Test beginnen. Zunächst wird eine formatierte Diskette eingelegt. An Port C4 wird der Wert 11h bei Maxilaufwerken mit einfacher Dichte ausgegeben, bei Minilaufwerken der Wert 21h für doppelte Dichte oder 31h für einfache Dichte (siehe *Abb. 7.4.14*). Danach wird der Wert 0Fh auf Port 0C0h ausgegeben. Damit wird ein Restore-Befehl ausgeführt. Das Laufwerk muß nun ansprechen (LED am Gehäuse leuchtet auf, falls vorhanden) und dann wieder ausgehen. Nun gibt es verschiedene Fälle nach Einlesen von Port 0C0H. *Abb. 7.4.14* zeigt ein Beispiel. Die Fehlerbelegung entspricht *Tabelle 7.4.7*. Hier also ein Beispiel, das o. k. ist. Bit 6 gesetzt bedeutet „Schreibschutz", da die Diskette schreibgeschützt war. Bit 5 bedeutet „Kopf liegt noch auf", da das Laufwerk noch selektiert war. Bit 2 auf 1 bedeutet „Spur 0", was auch o. k. ist, denn

ST 1
o o 6
o—o 5
o o 4 keine Präcompensation
o—o 3
o o 2
o—o 1

ST 3
Motor an mit Portbit 6

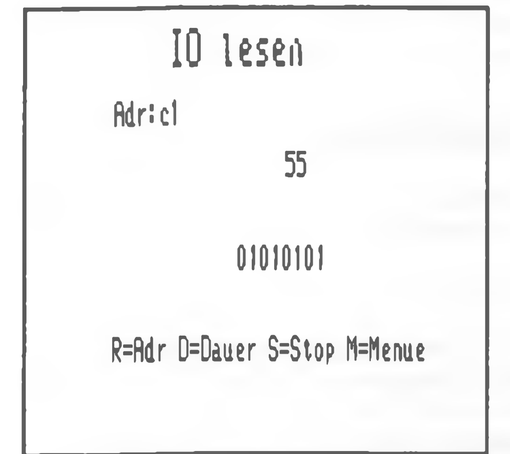
ST 6
o o—o Sideselect durch Software
Diese Brücke ist durch das Layout fest eingestellt.

ST 5
o o 7
o o 6 Adresse C0 eingestellt
o—o 5
o—o 4



Abb. 7.4.12 Ein Test mit dem Port C1, auf dem NDR-Klein-Computer gefahren

Abb. 7.4.11 (links) Die Belegung der Brücken auf der FLO2



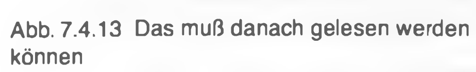


Abb. 7.4.13 Das muß danach gelesen werden können

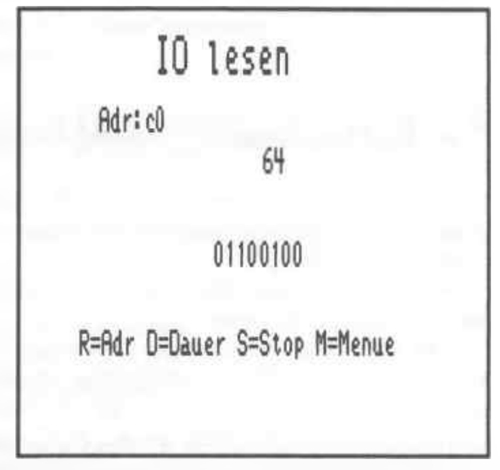


Abb. 7.4.14 Ein mögliches Ergebnis beim Restore-Befehl

der Befehl sollte ja die Spur 0 anfahren. Wenn man etwas später wieder abfragt, so erscheint der Wert 0 in allen Bits, denn das Laufwerk wird nur für eine bestimmte Zeit selektiert. Fehler werden signalisiert, wenn andere Bits gesetzt sind. Bit 7 würde zum Beispiel bedeuten „Laufwerk nicht Ready". Vielleicht steckt dann die Diskette falsch im Laufwerk, ihr „Label" muß zum Verschlußhebel zeigen (je nach Laufwerk). Bit 0 kann gesetzt bleiben, wenn z. B. der Indexpuls nicht kommt. Man kann das einmal probieren, indem man die Diskette aus dem Laufwerk herausnimmt und dann den Befehl 0Fh an Port 0C0h ausgibt. Wenn Bit 4 oder 3 gesetzt sind, so liegt ein

Lesefehler vor. Entweder stimmt dann die eingestellte Dichte nicht, oder die Diskette ist nicht formatiert (z. B. bei manchen käuflichen Minidisketten) oder die Schaltung arbeitet nicht einwandfrei. Man sollte dann neben einer Kontrolle der einzelnen Lötstellen auch die Diskette überprüfen. Aufschluß ergibt (für Experten) dann ein Oszillogramm der Signale RAWRD, RCLK am Controller-IC und $\overline{\text{RAWRD}}$ am Controller-IC (IC4). Auch könnte der Takt fehlen (XTAL-Pin 11 IC3, oder CLK Pin 24 IC4). Nun ein weiterer Test: Der Such-Befehl. Damit kann der Schrittmotor kontrolliert werden. Auf Port 0C4h gebe man wieder den Laufwerkscode (Maxi SD = 11h, Mini DD = 21h), auf Port 0C3h gebe man die Spurnummer, z. B. 20h und danach auf Portr 0C0h den Wert 1Fh, was dem Seek-Befehl entspricht. Jetzt muß das Laufwerk sich angesprochen fühlen und der Kopf sollte in eine innere Spur wandern. Anschließend kann man wieder den Status an Port 0C0h kontrollieren. Dort muß nach einigen Augenblicken der Wert 0 stehen oder, wenn man schnell genug mit der Abfrage ist, der Wert 20h oder 60h.

Zum Test unter dem 68008/68000 noch ein Hinweis. Beim 68008 werden die Adressen $FFFFFFC0 bis $FFFFFFC4 für den Controller verwendet und beim 68000 die Adressen $FFFFFFC0*2 bis $FFFFFFC4*2, da beim 68000 alle Systemports auf geraden Adressen liegen müssen.

Nachdem mit diesem Test auch das Lesen geprüft wurde, ist der Gesamttest schon fast beendet. Noch nicht getestet ist die Interrupt-Logik, die aber kaum Schwierigkeiten macht.

Jetzt müßte man das Betriebssystem laden können und loslegen. Beim 68000/8 Grundprogramm wähle man dazu das Menü Floppy-Start (ab Version 4.0). Beim mc-CP/M-Computer wird mit dem I-Befehl gearbeitet. Beim NDR-Klein-Computer erfolgt der Start mit Hilfe des neuen Programms FLOMON, das dazu auf der BANK/BOOT-Karte untergebracht ist. Für CP/M benötigt man beim Z80 64 KByte RAM und beim 68000/8 mindestens 128 KByte.

7.5 Aufbau eines EPROM-Programmierers

Wenn man eigene Programme in EPROMs festhalten möchte, braucht man dazu eine spezielle Baugruppe. Mit der hier beschriebenen Baugruppe lassen sich EPROMs vom Typ 2716, 2732 und 2764 programmieren.

Das Steuerprogramm dazu befindet sich bereits fertig im Grundprogramm und man braucht es nur über Menü aufzurufen.

Abb. 7.5.1 zeigt die Schaltung. Die Ausgabe der Daten erfolgt über das Latch IC7. Da es auch möglich sein muß, Daten von EPROM zurück zu lesen, ist der Treiber IC6 vorhanden. Der Ausgang des Latches, IC7, muß in den Tri-State-Zustand versetzt werden können, um Kollisionen zu vermeiden. Das geschieht über Pin 1, welches von einem weiteren Latch (IC9) über einen Inverter gesteuert wird. Die Adresse der EPROM-Zellen wird von Latch IC8 und einem Teil von IC9 angegeben.

Daten werden in das EPROM durch einen 50 ms langen Impuls eingeschrieben. Gleichzeitig wird eine erhöhte Spannung an einen der Pins gegeben (abhängig vom EPROM-Typ). Damit die Dauer des Programmierpulses immer gleich ist, wird dieser mit einem Monoflop, IC2, erzeugt.

Mit Tr1 muß man die Impulsdauer abgleichen.

Die Programmierspannung kann 25 V bzw. 21 V betragen, je nach EPROM-Typ. Als Spannung wird aber + 26 V oder + 22 V an die Baugruppe geführt, da ca. 1 V durch die Transistoren verlorengeht.

Am besten besorgt man sich vom Hersteller der EPROMs ein kostenloses Datenblatt mit den Programmierinformationen.

Abb. 7.5.2 zeigt die Belegung der einzelnen Ports. Die IO-Adressen sind von 80h bis 82h festgelegt. *Abb. 7.5.3* zeigt einzelne Wertbelegungen. Wenn man den Inhalt des EPROMs lesen will, so wird das Bit ena mit 0 belegt, -led mit 1 und trg mit 0. Also gibt man das Bitmuster 010xxxxx an den Port 81h. Dadurch erlischt die Leuchtdiode und das Latch IC7 gelangt in den Tri-State-Zustand. Damit ist das Monoflop nicht aktiv. An Port 80h kann man dann die Daten auslesen, wenn man an 81h die niederwertige Adressen A0 . . A7 legt und an 82h (anstelle der xxxxx-Bits) die Adressen A8 . . A12.

Beim Programmieren geht man anders vor. Nach Ausgabe der Datenbits an Latch IC7 und Ausgabe der Adressen A0 . . A7 an Port 81h gibt man nacheinander das Bitmuster 100xxxxx, 101xxxxx und dann 100xxxxx an Port 82h aus, wobei anstelle von xxxxx jedesmal die Belegung der Adreßbits A8 . . A12 stehen muß. Danach muß man 50 ms warten, man kann dies tun, indem man Bit 0 von Port 81h abfragt. Liegt das Bit auf 1, so muß man warten.

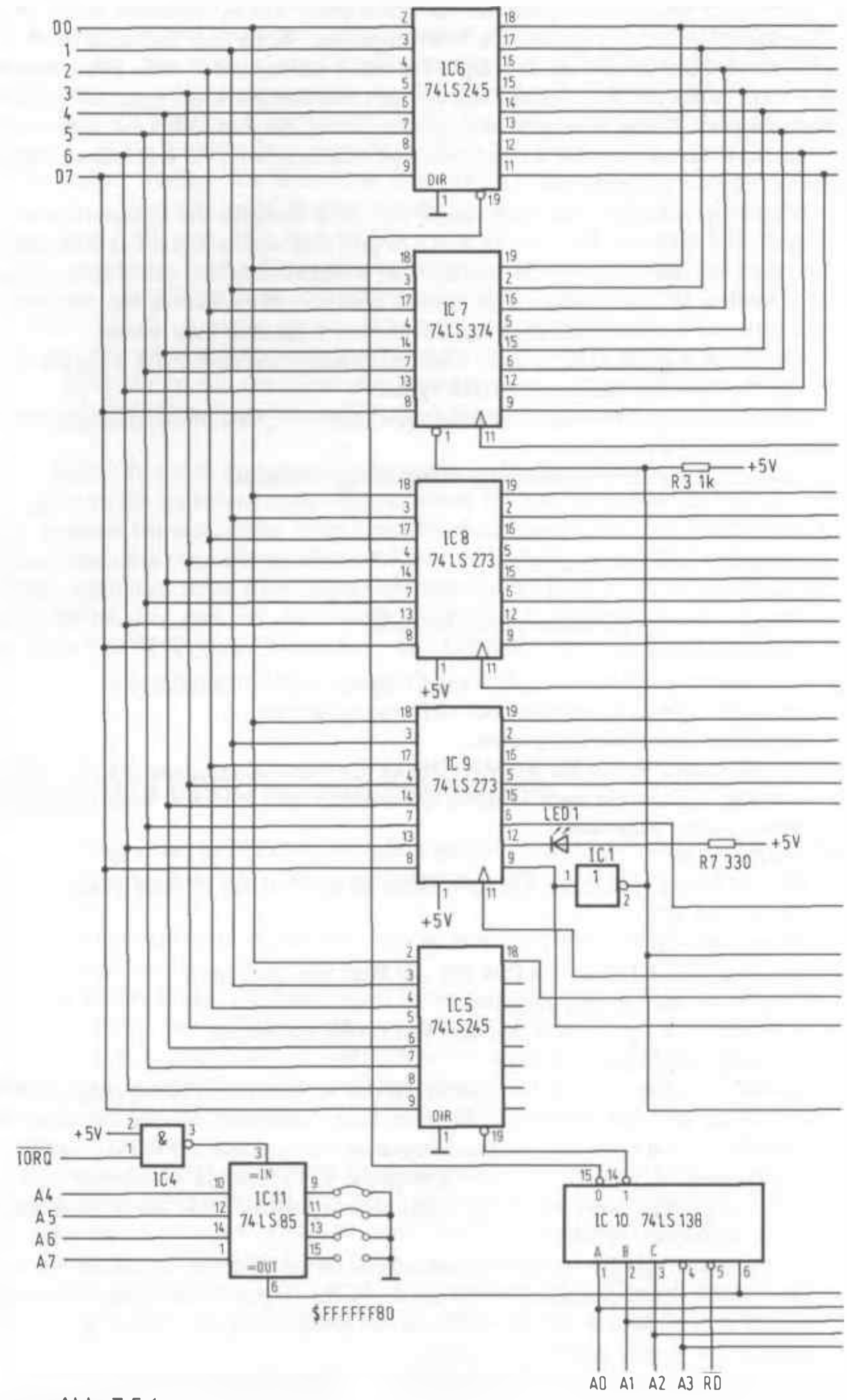
Abb. 7.5.4 zeigt die Belegung der Universalfassung, und *Abb. 7.5.5* zeigt die dazugehörigen Stecker für drei verschiedene EPROM-Typen.

Abb. 7.5.6 zeigt die Lötseite der Baugruppe, *Abb. 7.5.7* die Bestückungsseite und *Abb. 7.5.8* den Bestückungsplan.

Tabelle 7.5.1 zeigt schließlich die dazugehörige Stückliste.

Aufbau und Test der Baugruppe:

1. Einlöten aller Sockel und passiven Bauteile.
2. Einsetzen aller ICs.
4. Einsetzen in den Bus des NDR-Klein-Computers mit Grundprogramm.
4. Die + 26 V (+ 22 V) werden noch nicht angeschlossen.
5. Vortest mit dem Grundprogramm.
   Mit „IO setzen“ geben Sie den Wert 00h an den Port 82h aus, dann den Wert 40h an den Port 82h. Die LED müßte nach Eingabe des ersten Wertes leuchten, nach Eingabe des zweiten Wertes wieder ausgehen.
6. Weiterer Test.
   Mit „IO setzen“ geben Sie folgende Werte an den Port aus (h nicht eintippen):
   80h an Port 82h,
   55h an Port 80h.
   Jetzt muß bei „IO lesen“ an Port 80h der Wert 55h erscheinen.
   Dann 0AAh an Port 80h ausgeben.
   Jetzt bei „IO lesen“ an Port 80h der Wert 0AAh erscheinen.
7. Einstellen der Programmierzeit:
   Mit dem Trimmer Tr1 muß die Auslösezeit des Monoflops auf 50 ms eingestellt werden. Am einfachsten geht das, wenn man dazu ein Scop verwendet. Messen Sie dazu an Pin 6 des Monoflops. Nun rufen Sie im Grundprogramm die Funktion „EPROM programmieren“ auf. Als Startadresse wählen Sie 0, als Endadresse FFFF und als Zieladresse 0. Damit werden 64 KByte programmiert. An Pin 6 erscheinen nun positive Pulse, deren Breite Sie mit Tr1 auf 50 ms abgleichen müssen.
   Die Programmierpulse werden automatisch alle 60 ms ausgelöst, so daß der Abstand der Pulse gleichbleibt, wenn man eine Zeit kleiner als 60 ms eingestellt hat. Diese Verweilzeit, also 10 ms bei abgeglichenem Tr1, ist nötig, da der Kondensator am Monoflop eine gewisse Zeit braucht, um wieder geladen zu werden.
8. Nach dem Abgleich ist der PROMMER bereit für den großen Endtest.
   Besorgen Sie sich ein EPROM Ihrer Wahl, z. B. 2764 und vergessen Sie nicht den dazugehöri-


D0
1
2
3
4
5
6
D7
IC6
74LS245
DIR
IC 7
74LS 374
R 3 1k
+5V
IC 8
74 LS 273
+5V
IC 9
74 LS 273
LED 1
R7 330
IC1
+5V
IC5
74LS245
DIR
+5V
&
IC4
IORQ
A4
A5
A6
A7
IC 11
74 LS 85
=IN
=OUT
$FFFFFF80
IC 10 74LS 138
A
B
C
A0 A1 A2 A3 RD

zu Abb. 7.5.1

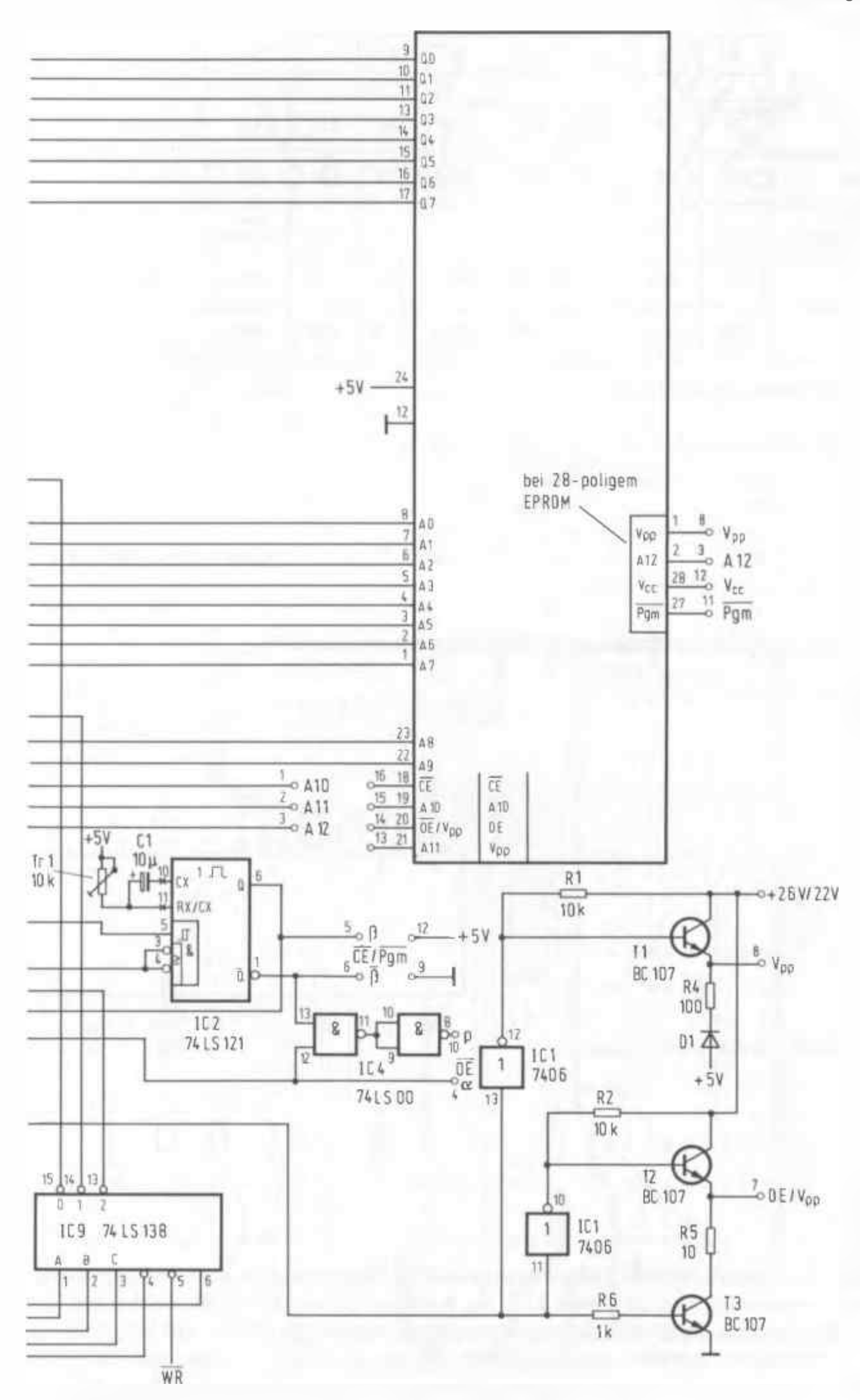


Abb. 7.5.1 Die Schaltung der Baugruppe PROMMER

Port	7	6	5	4	3	2	1	0	
80	D7	D6	D5	D4	D3	D2	D1	D0	Input
80	D7	D6	D5	D4	D3	D2	D1	D0	Output
81	x	x	x	x	x	x	x	busy 1=warten	Input
81	A7	A6	A5	A4	A3	A2	A1	A0	Output
82	ena (OE)	led 0=an	trg	A12	A11	A10	A9	A8	–

Abb. 7.5.2 Die Belegung der Ports

	ena (α)	led	trg (β)
Lesen	0	1	0
Programmieren	1 1 1	0 0 0	0 1 0

Abb. 7.5.3 Die Signale zur Programmierung

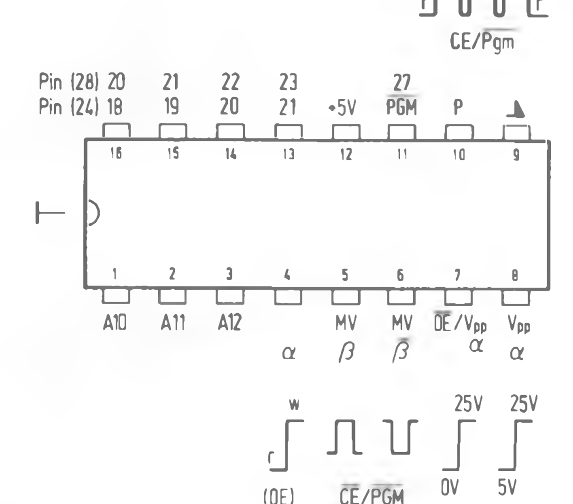
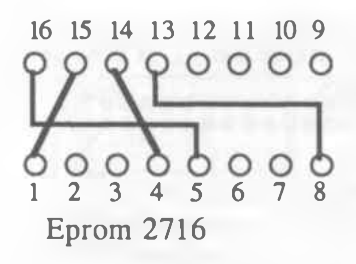
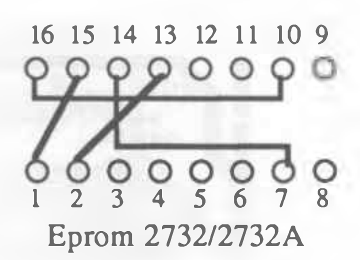


Abb. 7.5.4 Belegung der IC-Fassung

gen Stecker in den PROMMER einzusetzen. Schließen Sie auch die für das EPROM passende Programmierspannung von 26 V oder 22 V an. Achtung: Neuere EPROMs werden ggf. mit 12.5 V programmiert. Dafür braucht man dann eine Spannung von 13.5 V. Die Programmierspannung, die man an den PROMMER anlegt, muß generell um ca. 1 V höher sein, als die für das EPROM angegebene Programmierspannung, da ca. 1 V über den Transistoren abfällt. Die LED muß aus sein, dann dürfen Sie das EPROM einstecken. Also besser warten Sie, bis Sie im





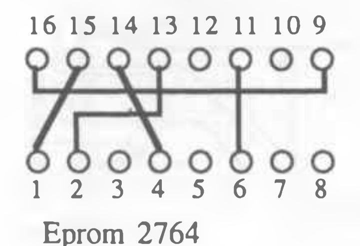


Abb. 7.5.5 Belegung von verschiedenen Steckern

Abb. 7.5.6 Lötseite der Baugruppe PROMMER

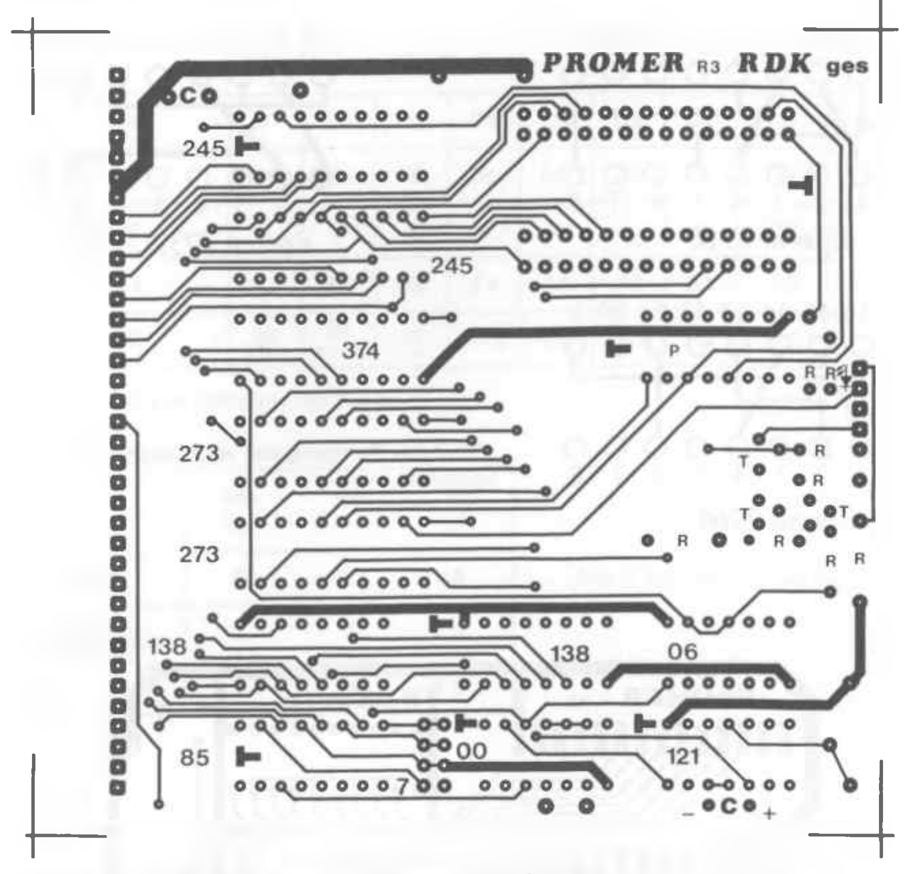


Abb. 7.5.7 Bestückungsseite der Baugruppe PROMMER

Menü „EPROM programmieren" sind. Nun kann man einmal ein paar Bytes programmieren. Dazu geben Sie als Startadresse 0 ein, dort sitzt das Grundprogramm und als Endadresse FF. Es werden also nur 256 Bytes programmiert, das soll fürs erste als Test reichen. Als Zieladresse geben Sie 0 ein, denn die Bytes sollen ab Adresse 0 in das EPROM geschrieben werden. Drücken Sie dann „B" für Bereit. Im unteren Bildschirmteil wird die aktuell programmierte Adresse ausgegeben.

Nach einer Weile meldet das Grundprogramm dann entweder einen Fehler oder ok. Im Fehlerfall müssen Sie die Schaltung sorgfältig kontrollieren.

Wenn alles ok war, können Sie für den Endtest das ganze EPROM programmieren. Geben Sie dazu 0 als Startadresse, 1FFF als Endadresse und 0 als Zieladresse ein. Der Programmiervorgang dauert einige Minuten. Die ersten schon programmierten Zellen werden hier einfach nochmals gebrannt, was aber nicht weiter stört.

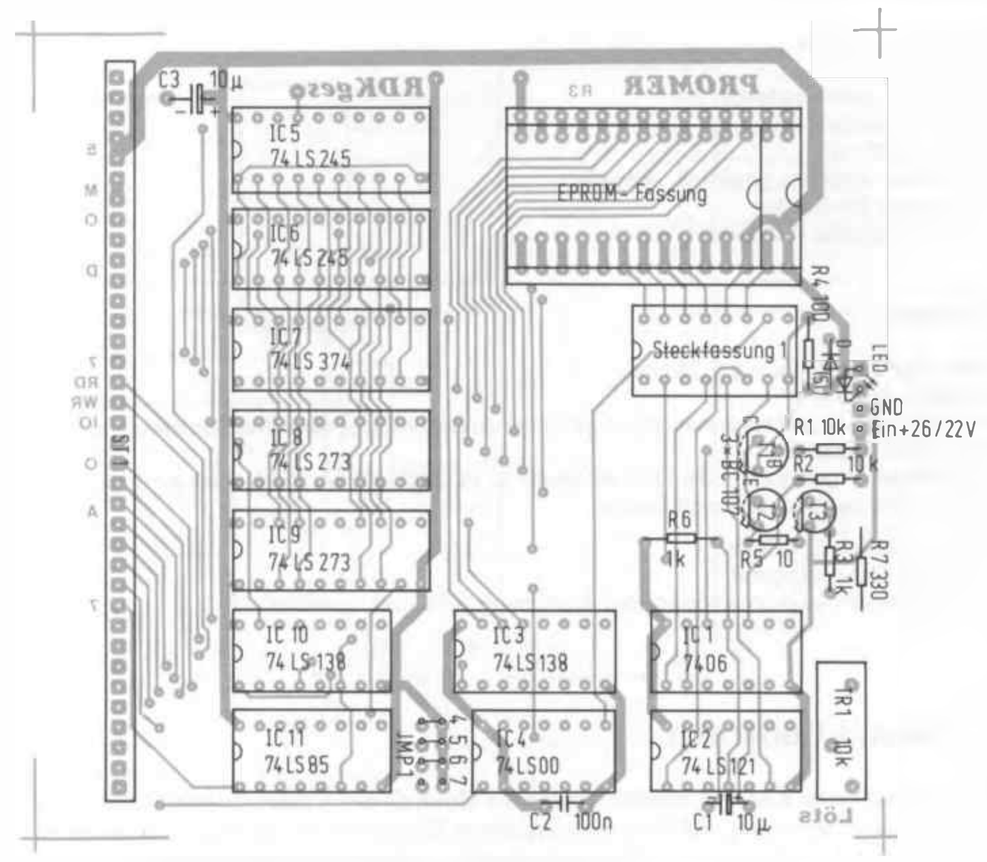


Abb. 7.5.8 Bestückungsplan der Baugruppe PROMMER

Tabelle 7.5.1 Stückliste zur Baugruppe PROMMER

IC1 7406 Inverter mit offenem Kollektor und hoher Kollektor-Spannung
IC2 74121 Monoflop
IC3, IC10 74LS138 1 aus 8 Dekoder
IC4 74LS00 Nand-Glieder
IC5, IC6 74LS245 bidirektionale Bustreiber
IC7 74LS374 Zwischenspeicher mit TriState-Ausgang
IC8, IC9 74LS237 Zwischenspeicher mit Löscheingang
IC11 74LS85 Vergleicher
1× 28polige EPROM-Fassung (Zero-Force-Typ)
5× 20polige IC-Fassung
4× 16polige IC-Fassung
3× 14polige IC-Fassung
R1, R2 10 kΩ 1/8 Watt
R3, R6 1 kΩ
R4 100 Ω
R5 10 Ω
R7 330 Ω
Tr1 10 kΩ Helitrimmer

C1, C3 10 µF, Tantal
C2 100 nF
LED1 3 mm Leuchtdiode rot
D1 Silizium Diode
T1, T2, T3 BC 107
St1 36polige Stiftleiste, gewinkelt, einreihig.
3× 16poliger DIL-Stecker
1× ges-Leiterplatte PROMMER

Kenndaten:

Spannungsversorgung: + 5 V, 280 mA
+ 26/22 V, ca. 40 mA
Die + 26 V oder + 22 V kann man einem gesonderten Netzteil oder einer Wandler-Baugruppe (POW22/26) entnehmen.
Der Prommer ist geeignet für die EPROM-Typen: 2716, 2732, 2732A, 2764, sowie für ähnliche Bauarten, bei geeignetem Anpaß-Stecker.

Zum Löschen der EPROMS:
1× UV-EPROM-Löschgerät.
Diese Geräte gibt es in unterschiedlicher Ausführung im Handel zu kaufen.

7.6 Sound-Generator

Hier wird eine kleine Karte beschrieben, die sowohl Musik als auch Geräusche produzieren kann. Dazu wird ein IC verwendet, das drei programmierbare Tongeneratoren sowie einen Rauschgenerator beinhaltet. Jeder Tongenerator kann in der Tonhöhe sowie Lautstärke programmiert werden. Ferner ist die Tonlage des Rauschgenerators programmierbar. Alle Ausgangssignale der Generatoren können gemischt werden und anstelle der Lautstärkeprogrammierung kann auch ein programmierbarer Hüllkurvengenerator verwendet werden.

Abb. 7.6.1 zeigt die Schaltung der Karte. Die Dekodierung erfolgt wie üblich mit einem Vergleicher, und der Bustreiber B1 dient zur Trennung des Datenbusses. Der Baustein benötigt ein paar ungewöhnliche Signale mit der Bezeichnung BDIR, BC1, die aber im Prinzip ähnlich wirken wie R/-W und -CS. Die Signale werden mit den Gattern NO1 und NO2 erzeugt. In BC1 ist außerdem noch die Adreßinformation enthalten. Das IC wird mit zwei Adressen angesprochen. Die untere Adresse, bei uns also 40h, führt an ein internes Adreßregister. Die darin enthaltene Adresse bewirkt die Auswahl eines von 16 internen Registern. Die Daten werden über 41h an das IC ausgegeben. Daten können auch aus den internen Registern gelesen werden, dies geschieht aber über die Adresse 40h.

Die Ausgänge A, B und C des Sound-Generators werden direkt zusammengeschaltet und am Sound-Ausgang hinter C1 liegt die NF-Spannung an. Sie kann z. B. an den Tonbandeingang eines Radios geführt werden oder an einen getrennt aufgebauten NF-Verstärker. Der Sound-Generator wird mit einer Frequenz von 2 MHz betrieben, dabei ergibt sich der beste Frequenz-Bereich der Tongeneratoren, z. B. für die Erzeugung von Melodien. Mehr als 2 MHz verträgt der Baustein nicht. Daher ist für ein 4 MHz-Z80-System auf der Karte noch ein Teiler mit FF1 aufgebaut. Über die Brücke J kann dann die Frequenz ausgewählt werden. Nun zu den internen Registern.

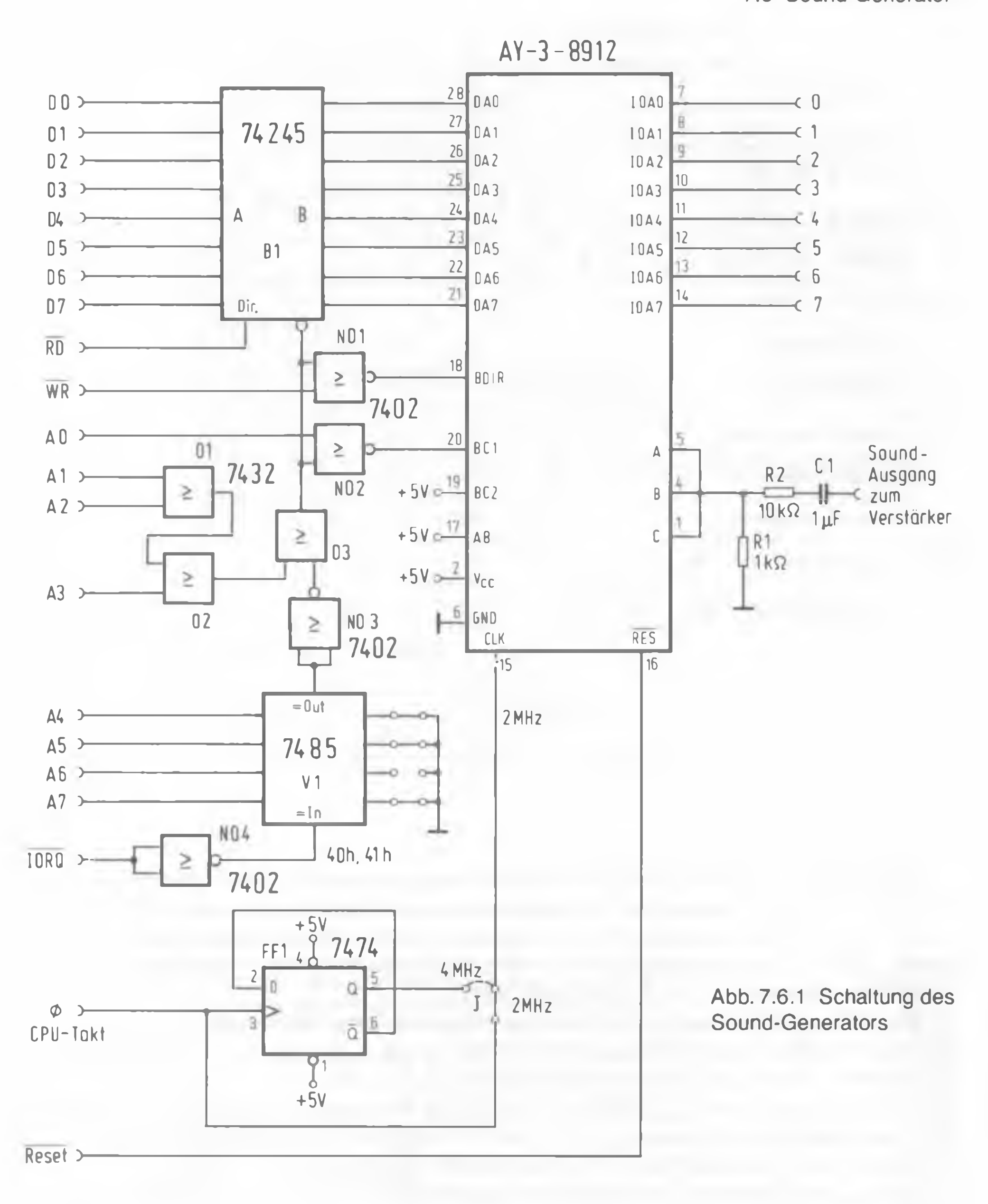


Abb. 7.6.1 Schaltung des Sound-Generators

Abb. 7.6.2 zeigt die Aufteilung. Mit den Registern R0 bis R5 wird die Tonhöhe der einzelnen Generatoren eingestellt. Jeweils zwei Register bestimmen einen Ton, da mit 12 Bit gearbeitet wird. Dabei wird der Eingangstakt zunächst einmal immer durch 16 dividiert. Dann wird durch herunterzählen eines 12-Bit-Zählers, der mit dem angegebenen Wert geladen wird, die Ausgangsfrequenz erzeugt.

Eine Rauschquelle ist mit Register R6 programmierbar. Die Grundfrequenz des Rauschgenerators wird durch herunterteilen der Taktfrequenz um 16 erreicht.

Reg		7	6	5	4	3	2	1	0
0	Kanal A Ton LSB	7	6	5	4	3	2	1	0
1	Kanal A MSB	x	x	x	x	B	A	9	8
2	Kanal B Ton MSB	7	6	5	4	3	2	1	0
3	Kanal B MSB	x	x	x	x	B	A	9	8
4	Kanal C Ton LSB	7	6	5	4	3	2	1	0
5	Kanal C MSB	x	x	x	x	B	A	9	8
6	Rauschperiode	x	x	x	x	3	2	1	0
7	Freigabe 0=an	In/Out (IOB)	In/Out (IOA)	Rausch C	Rausch B	Rausch A	Ton C	Ton B	Ton A
8	Kanal A Amplitude	x	x	x	M	L3	L2	L1	L0
9	Kanal B Amplitude	x	x	x	M	L3	L2	L1	L0
A	Kanal C Amplitude	x	x	x	M	L3	L2	L1	L0
B	Hullkurvenperiode LSB	7	6	5	4	3	2	1	0
C	Hullkurvenperiode MSB	F	E	D	C	B	A	9	8
D	Hullkurvenform	x	x	x	x	Cont	Att	Alt	Hold
E	I/O (Port A)	7	6	5	4	3	2	1	0
F	I/O (Port B) nur bei 8910	7	6	5	4	3	2	1	0

M=1, dann Hullkurve

Abb. 7.6.2 Register des Sound-Generators

Mit Register 7 können die einzelnen Quellen freigegeben werden. Eine 0 im entsprechenden Bit gibt sie frei. Im Soundgenerator ist aber auch noch ein Parallelport vorhanden. Die Richtung kann ebenfalls programmiert werden. 0 programmiert den Port als Eingang. Dabei gibt es im AY-3-8912, das ist der Baustein, den wir verwenden, nur einen Port, wohingegen der sonst kompatible Baustein AY-3-8910 zwei Ports enthält.

Die Amplitude der drei Kanäle (Ton oder Rauschen) kann mit den Registern R8 bis RA bestimmt werden. Dabei gibt ein Wert von 0 bis 15 die Lautstärke an. Die Lautstärke wird logarithmisch eingestellt. Zum Ausschalten eines Kanals wird der Wert auf 0 gesetzt. Ist das Bit 4 gesetzt, so wird die Lautstärkeeinstellung von einem Hüllkurvengenerator gesteuert. Die Hüllkurvenperiode läßt sich mit den Registern RB und RC einstellen, dabei wird mit 16 Bits gearbeitet, um auch Perioden mit sehr großer Zeitdauer erhalten zu können.

Mit Register RD kann die Hüllkurvenform eingestellt werden. *Abb. 7.6.3* zeigt die verschiedenen Hüllkurvenformen, die programmierbar sind. Die Hüllkurve wird beim Einschreiben in Register RD gestartet.

Register RE ist die direkte Verbindung zum I/O-Port. RF wird beim AY 9812 nicht verwendet.

Abb. 7.6.4 zeigt die Lötseite der Platine des Sound-Generators. In *Abb. 7.6.5* ist die Bestükkungsseite gezeigt und *Abb. 7.6.6* zeigt den Bestückungsplan.

Da die Schaltung nicht sehr umfangreich ist, ist der Aufbau recht einfach. Es werden passive Bauteile und Sockel eingelötet. Die Brücken werden so eingelötet, daß sich die Adressen 40h (41h) ergeben.

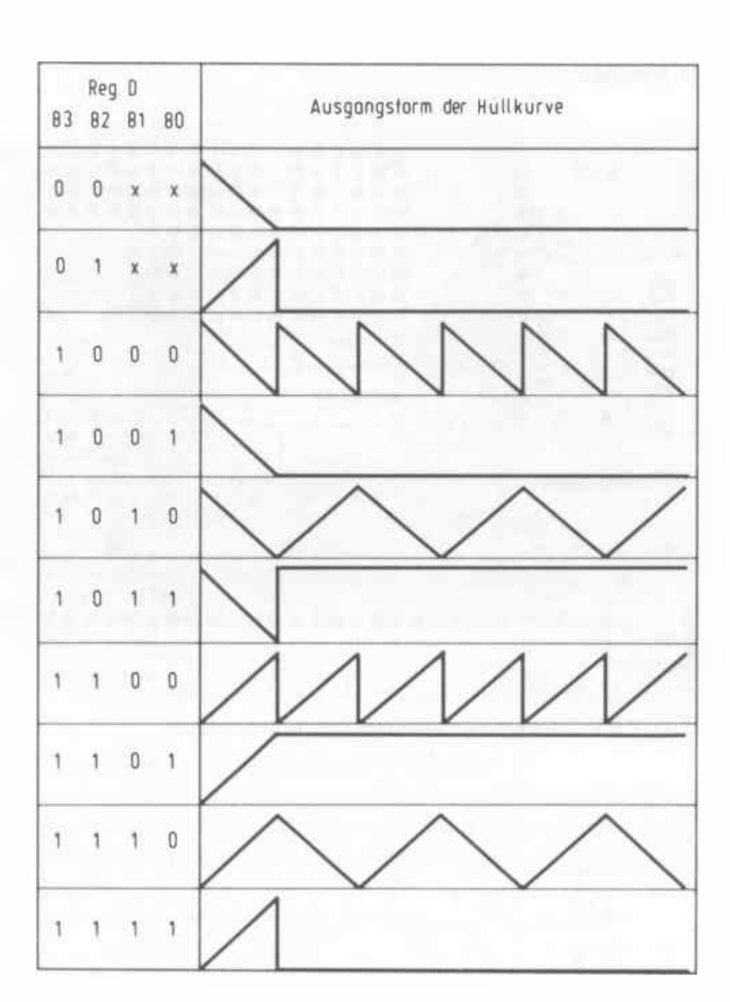


Abb. 7.6.3 verschiedene Hüllkurven

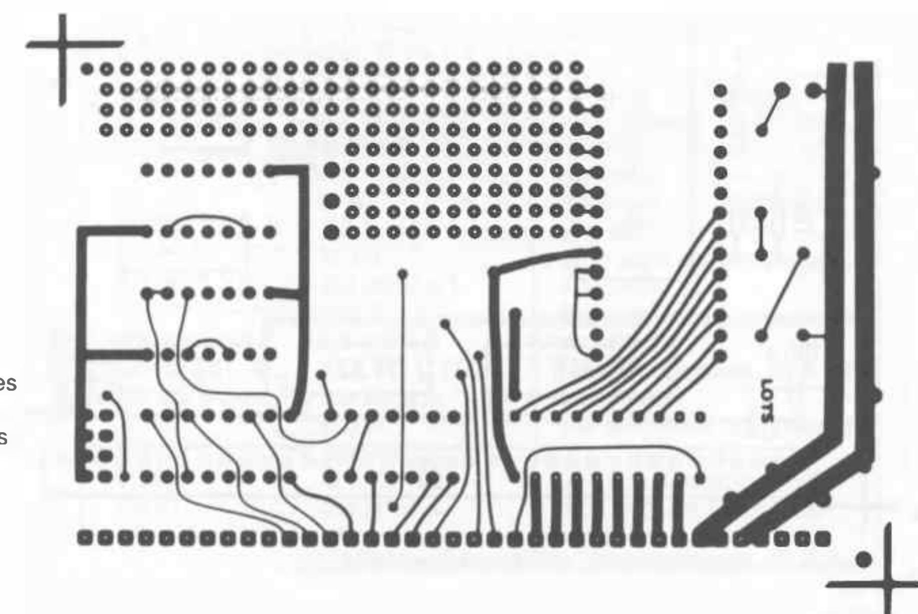


Abb. 7.6.4 Lötseite des Sound-Generators

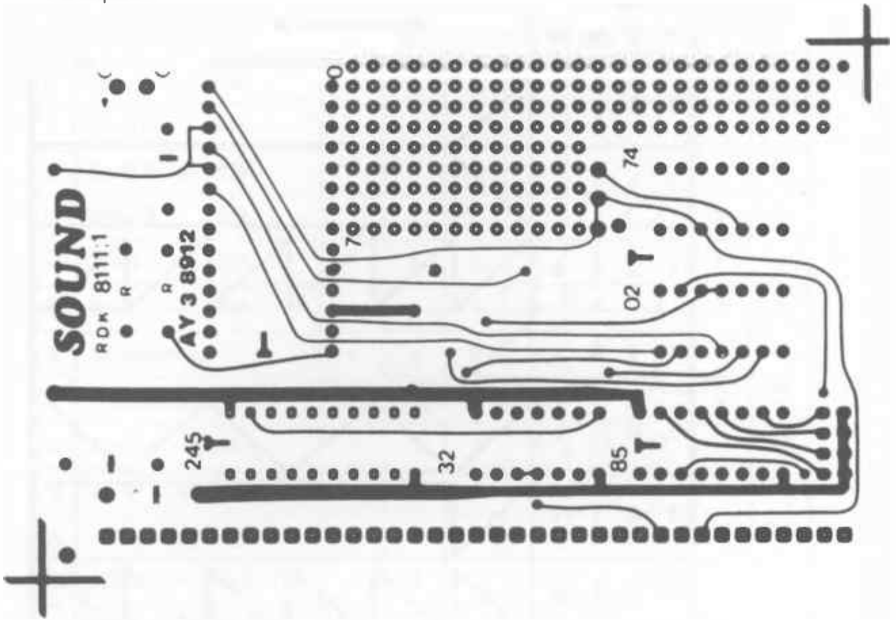


Abb. 7.6.5 Bestückungsseite des Sound-Generators

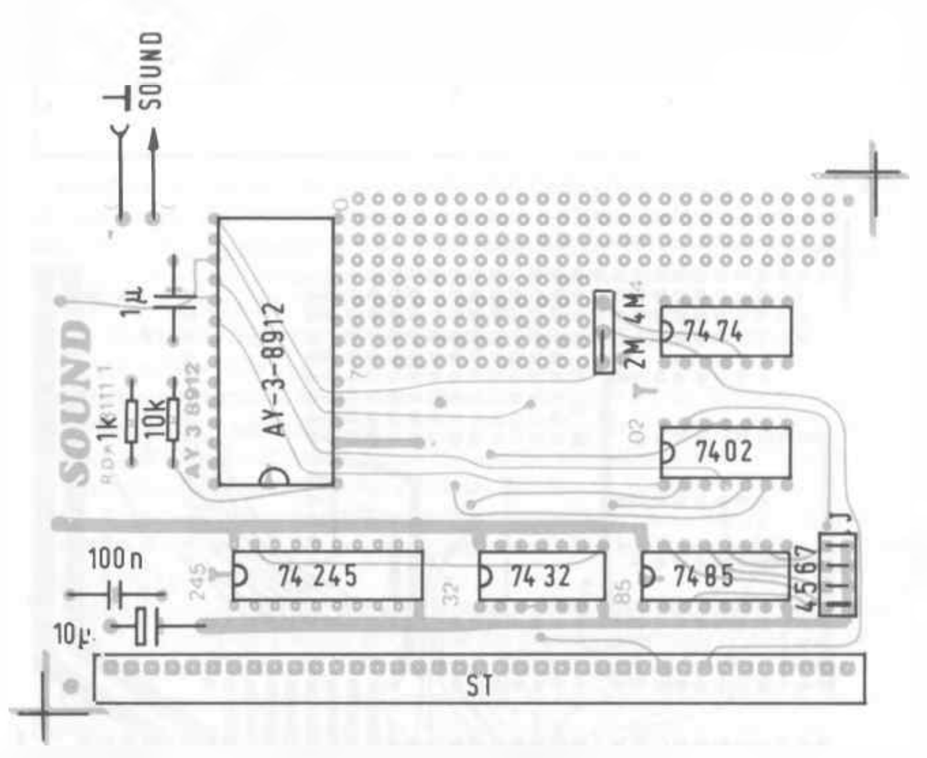


Abb. 7.6.6 Bestückungsplan des Sound-Generators

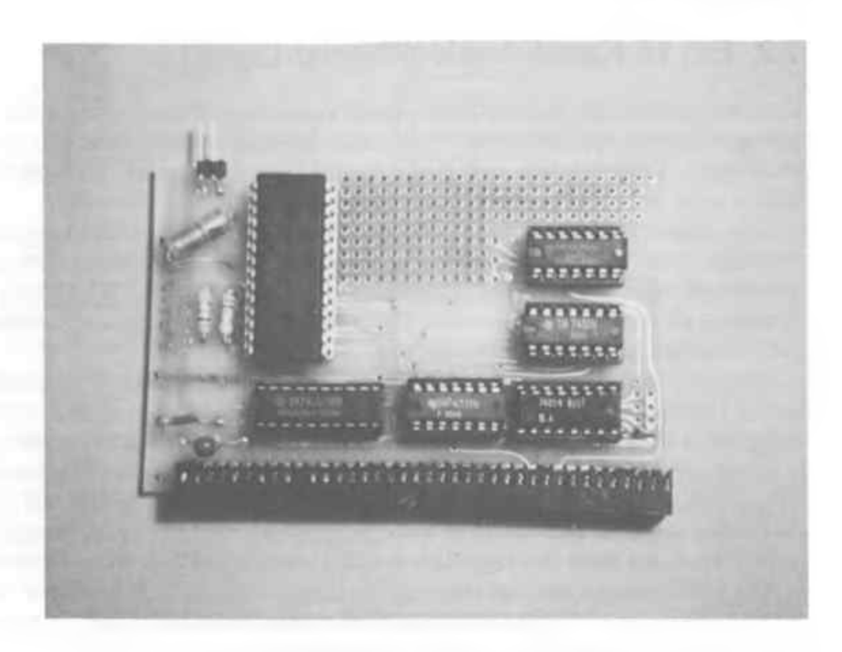


Abb. 7.6.7 Sound-Generator

1. Messen der Versorgungsspannungen, bevor die ICs eingesetzt werden.
2. Einsetzen aller ICs. Eingabe des Programms nach *Abb. 7.6.8*. Nach dem Start wird erst einmal die Dekodierung geprüft. Am Ausgang des Vergleichers V1 Pin 6 müssen Pulsgruppen erscheinen. An Pin 15 des Soundgenerators muß ein 2-MHz-Takt anliegen und Pin 16 des 8912 muß auf einem High-Pegel liegen.
3. Nun wird am Ausgang des Soundgenerators hinter dem Kondensator C1 gemessen.

Dort muß eine Frequenz mit einer Periode von etwa 700 µs anstehen. Dann ist die Schaltung in Ordnung.

Abb. 7.6.8 Test des Sound-Generators

Test des Soundgenerators			
3E 00	start:	ld a,0	Adresse 0 anwählen
D3 40		out (40h),a	ins Adresslatch ausgeben
3E 55		ld a,55h	Wert für Tonperiode LSB
D3 41		out (41h),a	ausgeben
3E 01		ld a,1	neue Adresse anwählen
D3 40		out (40h),a	und aktivieren
3E 00		ld a,0	Wert MSB der Tonperiode
D3 41		out (41h),a	ausgeben
3E 07		ld a,7	Freigabekanal
D3 40		out (40h),a	anwählen
3E FE		ld a,11111110b	Kanal A freigeben, Bit 0 = 0
D3 41		out (41h),a	ausgeben
3E 08		ld a,8	Die Amplitude anwählen
D3 40		out (40h),a	
3E 0F		ld a,15	und auf Maximum stellen
D3 41		out (41h),a	jetzt muß ein Ton hörbar
C9		ret	sein.

7.7 Ein 16-Kanal-Analog/Digital-Umsetzer

Computer können nur digitale Größen direkt verarbeiten. Wünschenswert ist es jedoch auch mit analogen Größen, wie Spannung, Widerstand, Strom oder Temperatur, Geräusch usw. umgehen zu können. Dazu braucht man einen Analog/Digital-Umsetzer. Er setzt eine Spannung in Zahlenwerte um, die man dann weiter verarbeiten kann.

Alle anderen analogen Größen kann man wieder relativ leicht in Spannungswerte überführen. Will man eine Temperatur messen, so verwendet man z. B. einen temperaturempfindlichen Widerstand oder besser einen integrierten Temperaturfühler (z. B. LM34), der gleich eine Spannung als Ausgangssignal liefert. Ggf. muß man die Spannung noch verstärken, um sie dem A/D-Umsetzer zuführen zu können.

Es gibt verschiedene Verfahren, Spannungen in digitiale Größen umzuwandeln, leider würde es den Umfang des Buches sprengen, hier näher darauf einzugehen. Wir verwenden hier einen integrierten Umsetzer, der das sogenannte sukzessive Approximationsverfahren verwendet. Es zeichnet sich durch eine hohe Umsetzrate aus, und der verwendete Baustein erreicht eine Wandelrate von ca. 100 kHz. *Abb. 7.7.1* zeigt die Schaltung. Der Baustein ADC0816 hat eine Auflösung von 8 Bit und besitzt 16 analoge Eingänge, die intern an einen analogen Multiplexer geführt sind, der dann den eigentlichen A/D-Umsetzer mit dem Eingangssignal versorgt.

Der A/D-Umsetzer benötigt einen eigenen Umsetztakt, der hier von einem Oszillator mit dem IC7413 erzeugt wird. Der Takt muß nicht besonders stabil sein und liegt um 1 MHz. Er wird dann noch durch zwei geteilt und als 500-kHz-Takt an den Wandler geführt.

Der Wandler belegt 16 IO-Adressen von E0 bis EF. Ein Schreibzugriff auf eine der 16 Adressen triggert den Wandler und startet den Umsetzvorgang.

Kurz nach der Wandlung geht das EOC-Signal des Wandlers auf Low, um dann nach der Wandlung wieder auf High zurückzugehen. Die Adresse, die man beim Schreiben wählt, bestimmt auch, welcher Kanal gewandelt wird. Also Adresse E0 wandelt den Kanal Vin 0, Adresse E1 wandelt den Kanal Vin 1 usw.

Das Signal EOC kann man per Prozessor abfragen, wenn man den Port E0 einliest. Bit 7 gibt dann den invertierten Status von EOC an. Nach der Umwandlung kann man den Datenwert an Port E1 einlesen.

Abb. 7.7.2 zeigt ein Testprogramm. Es wird zusammen mit dem Grundprogramm verwendet. Wenn man es startet, so erscheint eine horizontale Linie auf dem Bildschirm. Der Abstand der Linie vom unteren Rand entspricht der gemessenen Spannung. Wenn man an Vin 0 ein als Spannungsteiler geschaltetes Potentiometer anschließt, so kann man die Höhe der Linie direkt einstellen.

Bei Eingabe des Programms muß man noch auf eine Besonderheit des Grundprogramms achten. Die erste Zeile im Änderungsmenü lautet:

START: = $.

Damit wird dem Grundprogramm der Name START als Symbol genannt. Im Programm kann man später wieder auf dieses Symbol Bezug nehmen.

Der erste Programmschritt triggert den Wandler durch Ausgabe eines beliebigen Wertes an die Adresse E0.

Dann wird gewartet bis das Signal EOC auf Low geht, also am Port den Wert 1 annimmt. Danach wird in „loop2" darauf gewartet, bis das Signal EOC wieder auf High geht, also am Port den Wert 0 annimmt. Nun ist der Wandler bereit und man kann den Wert einlesen. Dies geschieht mit der Anweisung „in a,(0e1h)". Der Wert wird anschließend in das Register E befördert. DE bilden zusammen die X-Adresse und HL die Y-Adresse, wenn man den Befehl MOVETO oder

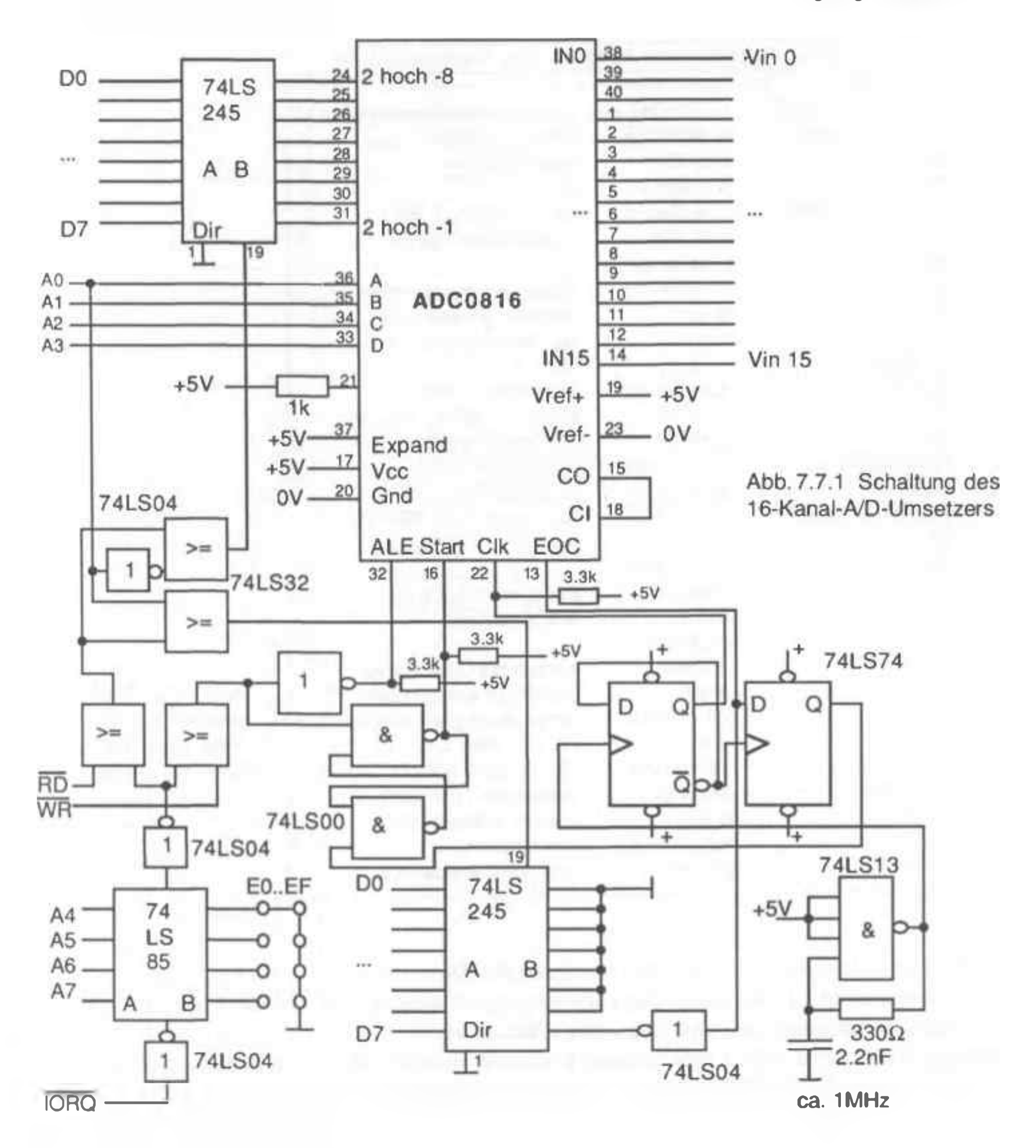


Abb. 7.7.1 Schaltung des 16-Kanal-A/D-Umsetzers

DRAWTO aus dem Grundprogramm verwendet. Damit läßt sich eine Linie zeichnen. In zwei weiteren Warteschleifen wartet das Programm auf den VSYNC-Impuls, um ein flimmerfreies Bild zu erhalten. Danach wird die Linie wieder gelöscht. Dann wiederholt sich der gesamte Meßvorgang. Dieses Programm kann man als Ausgang für verschiedene eigene Experimente nehmen.

1. Aufbau eines Zeigerinstruments.
2. Aufbau eines Oszilloskopes. Dazu muß man allerdings eine ganze Meßreihe speichern und ggf. mit zwei Bildseiten arbeiten. Man kann damit Signale bis zu 50 kHz erfassen (bei doppelter Abtastrate).

```
Testprogramm mit graphischer Ausgabe für's Grundprogramm
START:=$
D3 E0       START:  out (0e0h),a    Dummy Ausgabe startet A/D
DB E0       loop1:  in a,(0e0h)     warten bis EOC aktiv
E6 80               and 80h         geworden ist
28 FA               jr z,loop1      dann
DB E0       loop2:  in a,(0e0h)     warten warten, bis
E6 80               and 80h         EOC wieder inaktiv
20 FA               jr nz,loop2     ist.
DB E1               in a,(0e1h)     Wert kann eingelesen
5F                  ld e,a          werden. Er wird nach
16 00               ld d,0          DE transportiert = y
21 00 00            ld hl,0         HL = 0
CD MOVETO           call moveto     Startpunkt setzen
21 FF 01            ld hl,511       DE noch gültig, HL=511
D5                  push de         DE retten, sonst weg.
CD DRAWTO           call drawto     Linie zeichnen
D1                  pop de          DE wieder zurück
DB 70       loop3:  in a,(70h)      nun warten bis VSYNC
E6 02               and 2           weg ist
20 FA               jr nz,loop3
DB 70       loop4:  in a,(70h)      dann warten bis es
E6 02               and 2           wieder auftritt
28 FA               jr z,loop4
CD WAIT             call wait       warten bis GDP fertig
3E 01               ld a,1          Löschstift einschalten
D3 70               out (70h),a     damit dann die Linie löschen
21 00 00            ld hl,0         HL = 0, also X=0
CD DRAWTO           call drawto     DE ist noch gültig
CD WAIT             call wait       warten bis GDP fertig ist
3E 00               ld a,0          wieder Schreibstift
D3 70               out (70h),a     einschalten
C3 START            jp start        und alles wiederholen.
```

Abb. 7.7.2 Testprogramm mit graphischer Ausgabe für's Grundprogramm

3. Sprache aufzeichnen und wiedergeben (mit dem D/A-Umsetzer).
4. Temperaturverlauf über einen Tag aufzeichnen und wiedergeben (mit dem Scop-Programm, einem Temperatursensor und langsamer Abtastrate).
5. Analoger Joystick. Wenn man mehrere Kanäle als Eingang und ein Kreuzknüppelpotentiometer (z. B. vom Modellflugzeugbau) verwendet, kann man eine recht interessante Eingabe z. B. für Spiele bauen.

7.8 D/A-Umsetzer

Das Gegenstück zum Analog/Digital-Umsetzer ist der D/A-Umsetzer. Er wandelt eine digitale Größe in eine analoge Größe um.

Mit der hier vorgestellten Schaltung kann man einen 8-Bit-Zahlenwert in eine analoge Spannung im Bereich von ca. 0 . . 5 V umwandeln.

Abb. 7.8.1 zeigt die Schaltung. Verwendet wird eine integrierte Schaltung ZN428 von Ferranti. Dieses IC benötigt nur eine einzige Versorgungsspannung und ist außerdem sehr einfach anzuschließen. Da die Schaltung so einfach ist, sind hier gleich zwei D/A-Umsetzer vorgesehen.

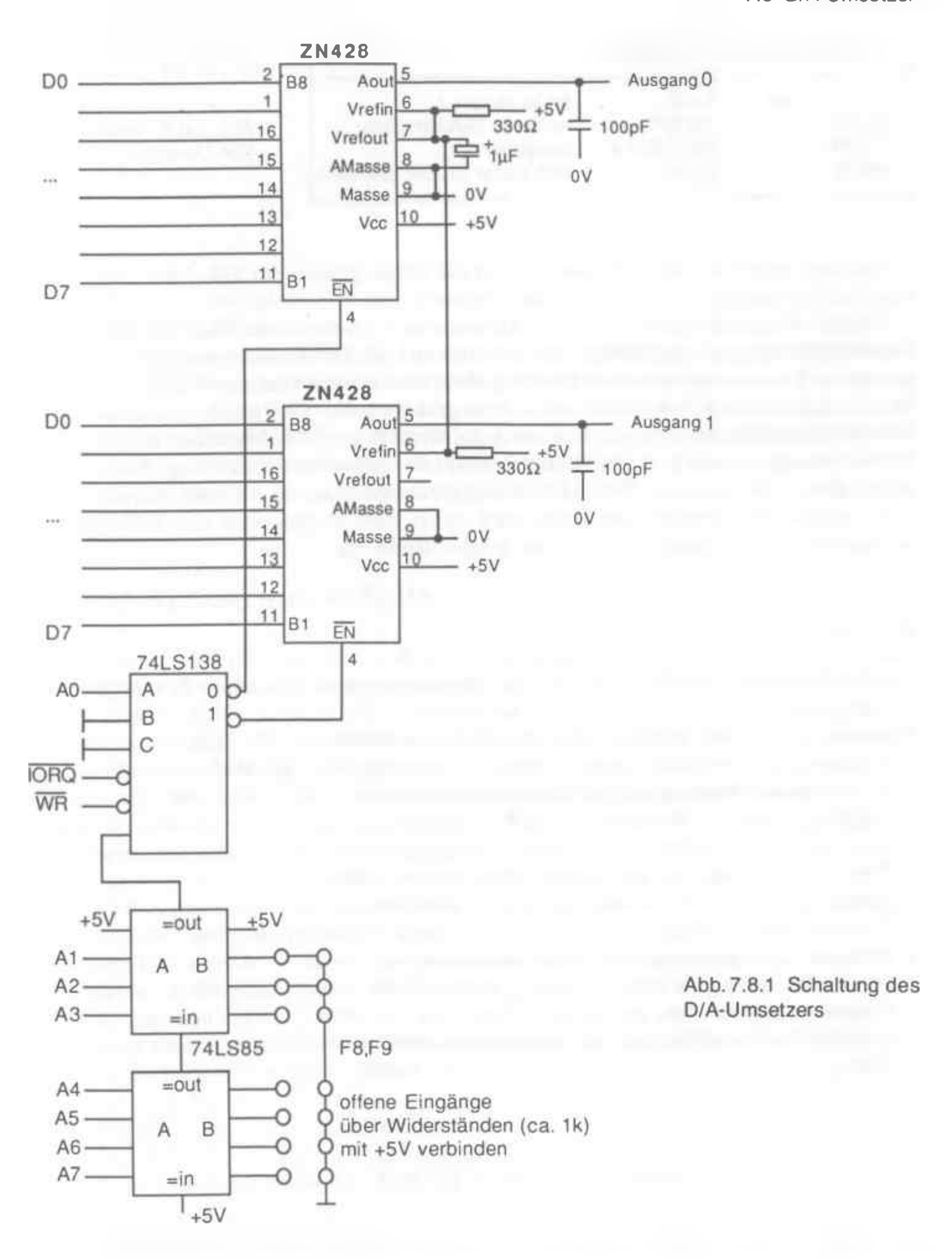


Abb. 7.8.1 Schaltung des D/A-Umsetzers

Die D/A-Umsetzer-Baugruppe belegt zwei I/O-Ports, nämlich F8h und F9h. Schreibt man z. B. einen Wert an die Adresse F8h, so wird er im dazugehörigen D/A-Umsetzer gespeichert und innerhalb von Mikrosekunden in einen analogen Spannungswert umgewandelt. Das Ergebnis liegt dann an Ausgang 0. Entsprechend ist der Port F9h dem Ausgang 1 zugeordnet.

Testprogramm D/A-Umsetzer			
3C	start:	inc a	Inhalt Akku + 1
D3 F8		out (0f8h),a	an beide D/A-Umsetzer
D3 F9		out (0f9h),a	ausgeben
18 F9		jr start	und weiter (0..255 zyklisch)

Abb. 7.8.2 Testprogramm D/A-Umsetzer

Genauso einfach wie die Wirkungsweise ist das Testprogramm, das *Abb. 7.8.2* zeigt, aufgebaut.

Der erste Befehl erhöht den Inhalt des Akkumulators A um eins. Dabei ist es hier egal, welcher Wert am Anfang darin stand. Dieser Wert wird dann an beide D/A-Umsetzer ausgegeben. Danach springt das Programm wieder an den Anfang. Jetzt wird A wieder um eins erhöht und der neue Wert ausgegeben. Der Wert des Registers A steigt also ständig. Dies geschieht solange, bis ein Überlauf stattfindet, wenn also das Register A den Wert 255 hat. Der nächste Wert ist dann wieder 0. Die Ausgangsspannung am D/A-Umsetzer steigt also immer kontinuierlich an, fällt dann aber wieder auf den Minimalwert. Man erhält eine Sägezahnspannung am Ausgang. Mit einem Scop kann man das auch sichtbar machen. Wer kein Scop hat, kann das Programm auch im Einzelschritt durchlaufen und mit einem Voltmeter die Ausgangsspannung messen.

Aufgaben:

1. Aufnahme einer Meßkurve. Messen Sie mit dem Voltmeter die minimale und maximale Ausgangsspannung der beiden D/A-Umsetzer.
2. Aufbau eines Funktionsgenerators. Ein Funktionsgenerator hat die Aufgabe, verschiedene Schwingungsformen zu erzeugen. Machen Sie dies, indem Sie eine Tabelle mit Funktionswerten verwenden, die dann an den D/A-Umsetzer ausgegeben wird. Verwenden Sie auch eine programmierbare Warteschleife, um die Periodendauer der Schwingung einstellen zu können. Die Ausgabe kann auch mit Hilfe eines Lautsprechers erfolgen, mit dem man verschiedene Töne und Klangarten der Schwingung hörbar machen kann.
3. Sprach- und Musikspeicherung und Ausgabe. Verwenden Sie dazu auch einen A/D-Umsetzer. Hinweise: man benötigt viel Speicher, um eine gute Qualität zu erreichen.
4. Erzeugen eines digitalen Echos. Man benötigt auch dazu einen A/D-Umsetzer (Kapitel 7.7). Die Werte müssen in einem großen Speicherbereich zwischengespeichert werden. Das Programm schreibt dann gleichzeitig Werte, die vom A/D-Umsetzer kommen, in diesen Speicherbereich und liest sie von einer anderen Stelle wieder aus und gibt sie an den D/A-Umsetzer.

8 Software

Dies Kapitel stellt eine Ergänzung zu den vorherigen Kapiteln dar. Wir werden darin im Abschnitt 8.1 die Z80-Befehle kennenlernen.

Im Abschnitt 8.2 sind die Befehle des Grundprogramms beschrieben. Eine kurze Zusammenstellung der Möglichkeiten mit dem Zeilenassembler zeigt Abschnitt 8.3. In Abschnitt 8.4 werden Sie eine einfache höhere Programmiersprache kennenlernen: Gosi (Logo-Teilmenge). Im Abschnitt 8.5 lernen Sie schließlich einen Basic-Interpreter kennen, und als Krönung des Ganzen finden Sie in Kapitel 8.6 den Flomon und eine kurze Einführung in das Betriebssystem CP/M und dessen Möglichkeiten.

8.1 Z80-Aufbau und Befehle

Bevor es an den Z80 geht, soll hier noch einmal eine kleine Wiederholung der Zahlensysteme durchgeführt werden. Dazu ein paar Beispiele. Die Zahl 123 soll in allen drei Systemen dargestellt werden. Erst einmal die Umrechnung in binär:

Die nächst kleinere Zweierpotenz (2 hoch n) ist 64, sie läßt sich von 123 abziehen und es bleibt 59 als Rest. Davon kann man 32 abziehen und es bleibt 27. Davon läßt sich 16 abziehen und es bleibt 11, 8 geht ebenfalls, Rest 3, 4 geht nicht, aber 2 und schließlich 1. Damit haben wir die Zahl zerlegt:

64 + 32 + 16 + 8 + 0 + 2 + 1

Im Binärsystem lautet die Zahl also

1111011

Wir wollen die Zahl nun oktal darstellen. Dazu geht man am besten von der Binärzahl aus und teilt sie von rechts nach links in Dreiergruppen auf:

001 111 011

Nun wird die jeweilige Dezimalzahl unter die Dreiergruppen geschrieben und es ergibt sich

173 als Oktalzahl.

Zur Umrechnung in HEX geht man ähnlich vor, die Binärzahl wird von rechts nach links in Vierergruppen aufgeteilt.

0111 1011

Und nun muß die entsprechende Dezimalzahl unter die Gruppe geschrieben werden:

7 11

11 ist aber eine dezimale Darstellung. Beim HEX-System benötigen wir weitere Zahlen, man hat sich auf die Buchstaben A bis F geeinigt und nimmt sie hinzu, so daß gilt:

1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

dann ergibt sich die Zahl

7B

als Hexzahl der dezimalen Zahl 123.

Um künftig die einzelnen Zahlendarstellungen auseinanderhalten zu können, wird hinter die Zahl eine Kennung geschrieben, dabei wird für das dezimale System keine Kennung verwendet, in Zweifelsfällen jedoch ein kleines d. Beim oktalen System, das wir jedoch nicht verwenden, wäre das Zeichen „o" die Kennung, beim binären System wird ein kleines „b" verwendet und beim hexadezimalen (sedezimalen) System wird ein kleines „h" gebraucht. Damit ergibt sich für das obere Beispiel

123 = 123d = 1111011b = 173o = 7Bh

Weiteres Beispiel

16333 = 16333d = 11111111001101b = 37715o = 3FCDh

Die Umrechnung ins dezimale System ist auch sehr einfach. Dazu folgendes Beispiel:

4BAh ist ins dezimale System umzurechnen.

Es gilt

4*16*16 + B*16 + A

als dezimale Darstellung.

Was aber mit B*16? Dazu wird die Zahl dezimal umgerechnet und B ist 11.
Die Zuordnung lautet:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Somit ist

4BAh = 4*16*16 + 11*16 + 10 = 1210d

die Lösung der Aufgabe.

Wir kommen jetzt zum Innenleben des Z80. Er besitzt eine Vielzahl von Speichereinheiten, die Register genannt werden. In solch einem Register können zum Beispiel die Zwischenwerte von

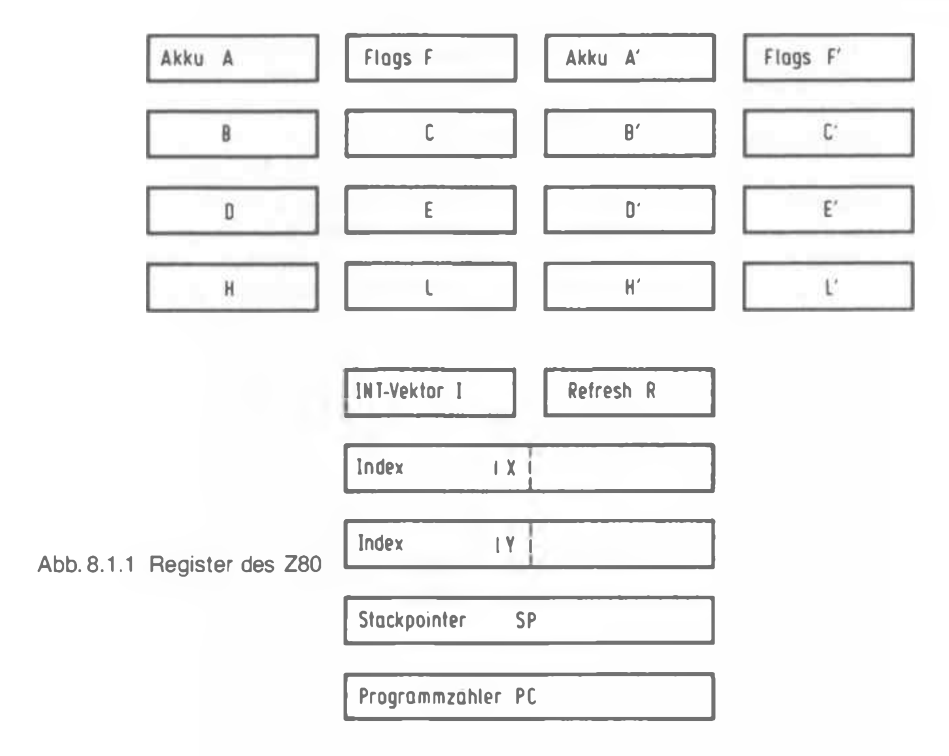


Abb. 8.1.1 Register des Z80

Rechnungen abgelegt werden. Manche der Register können auch Adressen aufnehmen, um den externen Speicher anzusprechen. Eines dieser Register ist der uns schon bekannte Programmzähler. *Abb. 8.1.1* zeigt das Innenleben des Z80. Durch die vielen Register darf man sich nicht verwirren lassen. Zwei Register sind für uns zunächst bedeutend. Einmal der schon bekannte Programmzähler, der mit PC bezeichnet ist und dann ein Register mit der Bezeichnung Akku A. Der Akku ist bei Rechnern meist ein besonderes Register. In ihm können Rechnungen, aber auch logische Verknüpfungen durchgeführt werden. Im Prinzip würden wir mit diesen beiden Registern, dem Akku und dem Programmzähler, auskommen. Tatsächlich gibt es andere Mikrocomputer, die praktisch nur diese beiden Register besitzen. Beim Z80 wird aber mindestens noch ein Register zum sinnvollen Arbeiten benötigt, wir wählen das Register B hinzu. Es wird benötigt, um mit dem Akku zusammenzuarbeiten (wir hätten auch eines der anderen Register C, D, E, H oder L verwenden können, aber keines der restlichen).

Nun fangen wir an, ein paar Befehle zusammenzustellen. Eine wichtige Gruppe sind die Ladebefehle. Sie haben die Aufgabe, einen Wert aus dem Speicher z. B. in ein Register zu laden. Man hat sich darauf geeinigt, von einem Ladevorgang zu sprechen, wenn vom Speicher in den Mikrorechner transportiert wird und von wegspeichern, wenn vom Mikrorechner in den Speicher transportiert wird, oder allgemeiner, Laden beim Transport von einer Umgebung in einen spezielleren Teil und Speichern von einem speziellen Teil in eine größere Umgebung. Die Definition ist aber nicht so streng zu sehen.

Der Z80 kann 64 KByte Speicher, also 65536 Speicherzellen mit je 8 Bit Datenbreite adressieren. Der Ladebefehl kann von einer dieser Zellen den Inhalt in das Register A transportieren. Wie wird der Ladebefehl nun angegeben? Der Befehl wird dem Z80 als Operations-Code

zugeführt, ähnlich wie in dem Beispiel der Verkehrsampelsteuerung aus Kapitel 5. Damit besitzt er eine binäre Darstellung:

001110101llllllhhhhhhhh

wobei die Abkürzung l für den niederwertigen Adreßteil steht und h für den höherwertigen Adreßteil. Doch nun werden Sie sicher bemerken, daß sind ja nicht 8 Bits, sondern 24. Wie geht das bei einem 8 Bit-Rechner? Ganz recht. Der Operationscode ist tatsächlich 24 Bit lang, doch wird er, da er sonst nicht auf den Datenbus des Z80 paßt, in drei 8-Bit-Gruppen zerlegt. Das sieht dann so aus:

00111010
llllllll
hhhhhhhh

Schon besser! Doch, was hat es mit der höher- und niederwertigen Adresse auf sich? Um die 65536 Zellen zu adressieren, wird eine 16-Bit-Adresse benötigt. Diese Adresse muß in zwei Teile zerlegt werden. Der Z80 will dabei zuerst die untere Hälfte der Adreßbits haben und dann die obere Hälfte. Beispiel: Es soll von der Zelle 4567h in den Akku geladen werden. Die Adresse 4567h ist binär 0100010101100111b. Also ergibt sich

00111010
01100111
01000101

Nun ist diese binäre Schreibweise nicht gerade lesbar und daher wird im allgemeinen die hexadezimale Schreibweise bei der Angabe von Befehlscodes verwendet. In Hex lautet das Ganze:

3A
67
45

Nun ist auch das noch nicht sehr lesbar, man stelle sich ein langes Programm vor, das nur aus Zahlenreihen besteht, wer soll sich merken und wissen, was dabei passiert? Daher hat sich jemand die mnemotechnische Darstellung von Operationscodes überlegt.
Der Befehl wird dann geschrieben:

LD A, (4567h)

Und nun wird die Funktion deutlich. LD steht für LOAD oder LADE, A steht für den Akku A und (4567h) steht für den Inhalt von 4567h, wobei ( ) immer als „Inhalt von" gelesen wird. Zum Schreiben der Programme wird die mnemotechnische Version verwendet und vor der Eingabe in den Computer wird die Darstellung in die HEX-Darstellung übersetzt. Das Übersetzen kann dabei von Hand geschehen, so wie wir es üben werden, oder es kann auch mit Hilfe des Computers selbst durchgeführt werden, der dazu ein Programm, den Assembler benötigt. Nun können wir uns einen Grundbefehlssatz zusammenstellen, der etwas lesbarer ist, als nur die reine Binärform.

Grundbefehlssatz:

Dazu folgende Konventionen:
adresse ist eine Abkürzung dafür, daß hier eine beliebige 16-Bit-Zahl stehen darf
data eine 8-Bit-Zahl (0..255) darf angegeben werden
Neben der allgemeinen Form ist noch ein Beispiel mit Codierung angegeben.
Alle Codeangaben sind auch ohne die Angabe des Zeichens h in HEX.

```
LD A,(adresse)      LD A,(1234h)      3A 34 12
```

der Akkumulator A wird mit dem Inhalt der Zelle (adresse) geladen.

```
LD (adresse),A      LD (1234h),A      32 34 12
```

Der Inhalt des Akkumulators A wird auf die Adresse (adresse) abgespeichert. Dies ist die genaue Umkehrung des vorherigen Befehls.

```
LD A, B             LD A, B           78
```

Der Inhalt des Registers B wird in den Akkumulator A geladen. Der Operationscode ist nur 1 Byte lang.

```
LD B,A              LD B,A            47
```

Der Inhalt des Akkumulators A wird in das Register B gespeichert. Hier ist der Operations-Code nur ein Byte lang.

```
LD A,data           LD A,5            3E 05
```

In den Akkumulator wird der Wert data geladen.

```
LD B,data           LD B,5            06 05
```

In das Register B wird der Wert data geladen.
Nun folgen ein paar arithmetische Befehle:

```
ADD A,data          ADD A,1           C6 01
```

Der Inhalt des Akkumulators A wird um data erhöht. Der Operationscode ist ein Zwei-Byte-Befehl.
Im Beispiel wird Register A um eins erhöht.

```
ADD A,B             ADD A,B           80
```

Zum Inhalt des Akkumulators A wird der Inhalt des Registers B addiert.

```
SUB data            SUB 1             D6 01
```

Vom Inhalt des Akumulators wird der Wert data subtrahiert.

```
SUB B               SUB B             90
```

Vom Inhalt des Akkumulators A wird der Inhalt des Registers B subtrahiert.

Wir haben schon früher gelernt, daß mit den obigen Befehlen nur ein lineares Programm aufgebaut werden kann. Wir wollen aber auch den Programmzähler verändern können und dazu gibt es die sogenannten Sprungbefehle.

```
JP adresse          JP 1200h          C3 00 12
```

Der nächste Befehl nach diesem wird von Adresse 1200h geholt. Damit wurde ein Programmsprung ausgeführt.

Es genügt nicht, nur springen zu können, dies muß auch von einer Entscheidung abhängig gemacht werden können. Dazu gibt es bedingte Sprungbefehle.

```
JP Z,adresse        JP Z,1200h        CA 00 12
```

Die Bedingung liegt in der Abfrage eines Bits, das als Null-Flag bekannt ist. Es wird bei arithmetischen Befehlen verändert und gibt an, wann der Inhalt eines Registers 0 wurde. Wurde der Akku A aufgrund einer Subtraktion 0, so wird das Null-Flag (Zero-Flag) gesetzt. Folgt daraufhin der obige Sprungbefehl, so wird er ausgeführt, wurde der Akku zuvor nicht 0 gesetzt, so ist das Flag (oder Merker) nicht gesetzt und der Sprung wird nicht ausgeführt.

```
JP NZ,adresse       JP NZ,1205        C2 05 12
```

Hier wird der Sprung ausgeführt, wenn das Null-Flag nicht gesetzt war; das Verhalten ist also genau umgekehrt wie beim vorherigen Befehl.

Mit diesen Befehlen können wir nun ein paar Programme erstellen.

Es sollen zwei Zahlen addiert werden. Die beiden Zahlen stehen im Speicher. Die erste Zahl steht auf Adresse 1305h und die zweite Zahl soll auf 1306h stehen. Zum Addieren müssen die beiden Zahlen zunächst in die Register transportiert werden, da eine Addition nur innerhalb der Register möglich ist.
Es gilt also:

```
LD A,(1305h)        ; laden erster Operand
LD B,A              ; transport nach B
LD A,(1306h)        ; laden zweiter Operand
```

Einen Befehl LD B,(adresse) gibt es beim Z80 leider nicht. Dann wird die Addition durchgeführt:

```
ADD A,B
```

und nun muß das Ergebnis noch abgespeichert werden. Dies kann mit dem Befehl

```
LD (1307h),A
```

durchgeführt werden. Das Resultat wird dann in Speicherzelle 1307h zu finden sein.

Das ganze Programm sieht dann so aus:

```
LD A,(1305h)
LD B,A
```

```
LD A,(1306h)
ADD A,B
LD (1307h),A
```

Nun ist dieses Programm aber für den Rechner noch nicht verständlich. Es muß dazu in Maschinensprache umgesetzt werden. Das Programm soll dabei auf Adresse 1200h beginnen. Die Umcodierung kann mit Hilfe der Liste von vorher durchgeführt werden und es ergibt sich:

```
1200    3A 05 13    LD A,(1305h)
1203    47          LD B,A
1204    3A 06 13    LD A,(1306h)
1207    80          ADD A,B
1208    32 07 13    LD (1307h),A
120B    . . . . . . . . . . . . . . . . . . . . . . . . .
```

Im Speicher wird die Sequenz dann in aufeinanderfolgenden Zellen abgelegt:

```
1200h:  3A  05  13  47  3A  06  13  80  32  07  13
```

Kommt der Prozessor zur Adresse 1200h, so wird unser Programm ausgeführt.

Nun die Anwendung einer unbedingten Anweisung. Wir wollen die Verkehrsampelsteuerung nachbilden. Dazu benötigen wir eine Verbindung mit der Außenwelt. Der Z80 besitzt dafür eigene Befehle, die IO-Befehle:

```
IN A,(data)          IN A,(41h)          DB 41
```

Der Datenwert des Ports wird in den Akkumulator A geladen. Hier wird nur eine 8 Bit-Adresse angegeben, da der Z80 nicht mehr als 256 IO-Adressen ansprechen kann.

```
OUT (data),A         OUT (54h),A         D3 54
```

Der Inhalt des Akkumulators A wird auf die Adresse 54h ausgegeben.

Mit diesen IO-Befehlen läßt sich die Steuerung realisieren. Dazu wird an einen Port, z. B. der mit Adresse 30, die Verkehrsampel angeschaltet. Ein Port besitzt beim Z80 im allgemeinen 8 Datenbits, wir benötigen aber nur drei bei einer Ampel. *Abb. 8.1.2* zeigt den Anschluß. Dem Bit 0 ist das rote Signal zugeordnet, Bit 1 erhält das Signal Gelb und Bit 2 wird an Grün angeschlossen. Soll z. B. Gelb leuchten und die anderen nicht, so muß am Ausgang des Ports Bit 1 auf 0 sein, die anderen auf 1. Null daher, weil die LEDs, die hier als Ausgabe verwendet wurden, mit der Katode am Ausgang liegen und daher nur dann leuchten, wenn sich ein 0V-Pegel am Ausgang befindet. Als erstes legen wir uns also die Bitmuster für die einzelnen Fälle zurecht.

```
ROT:          xxxxx110b
GELB:         xxxxx101b
GRÜN:         xxxxx011b
ROTGELB:      xxxxx100b
```

Nun muß der Ablauf des Programms festgelegt werden. Es soll folgen: ROT – ROTGELB – GRÜN – GELB – ROT – ROTGELB . . . . . Um diese Sequenz zu erhalten, müssen die einzelnen Bitmuster nacheinander an den Port gelegt werden.

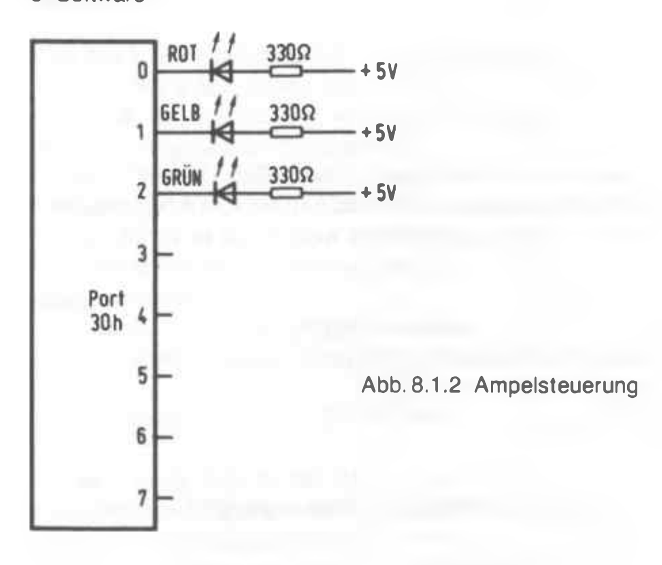


Abb. 8.1.2 Ampelsteuerung

Dazu wird das Bitmuster zunächst in den Akku A geladen und dann mit dem Ausgabebefehl an das Port ausgegeben. Die ganze Sequenz soll sich dann wiederholen. Dazu verwenden wir den Sprungbefehl. Das Zeichen x steht für beliebig, da die restlichen Bits nicht verwendet wurden. Wir können es z. B. mit 0 belegen.

```
LD A,00000110b          ; Lade ROT
OUT (30h),A             ; Ausgabe ROT
LD A,00000100b          ; Lade ROTGELB
OUT (30h),A             ; Ausgabe ROTGELB
LD A,00000011b          ; Lade GRÜN
OUT (30h),A             ; Ausgabe GRÜN
LD A,00000101b          ; Lade GELB
OUT (30h),A             ; Ausgabe GELB
JP zurück               ; Rücksprung
```

Bei dem Sprungbefehl steht einfach JP zurück. Wohin zurück, ist hier die Frage. Dazu muß eine Marke angegeben werden, man nennt dies auch LABEL. Die Marke trägt die Bezeichnung „zurück“.
Das Programm sieht dann wie folgt aus:

```
zurück:     LD A,00000110b      ; Lade ROT
            . . . wie vorher. . .
            JP zurück
```

Nun ist klar, wohin gesprungen werden soll. Der Doppelpunkt hinter der Marke „zurück“ dient dabei als Kennung und macht deutlich, daß hier von irgendwoher hingesprungen werden soll. Nun muß das Ganze noch kodiert werden. Wir wollen das Programm nun einmal auf Adresse 0 anfangen lassen.

```
0000  3E  06         zurück:   LD A,00000110b     ; Lade ROT
0002  D3  30                   OUT (30h),A        ; Ausgabe ROT
0004  3E  04                   LD A,00000100b     ; Lade ROTGELB
0006  D3  30                   OUT (30h),A        ; Ausgabe ROTGELB
0008  3E  03                   LD A,00000011b     ; Lade GRÜN
000A  D3  30                   OUT (30h),A        ; Ausgabe GRÜN
000C  3E  05                   LD A,00000101b     ; Lade GELB
000E  D3  30                   OUT (30h),A        ; Ausgabe GELB
0010  C3  00  00               JP zurück          ; Rücksprung
```

Als Sprungadresse wird hier der Wert 0000 eingesetzt. Würde das Programm auf Adresse 1234h starten, so wäre die Adresse der Marke „zurück" 1234h und die Sequenz wäre dann:

```
1234  3E  06         zurück:   LD A,00000110b     ; Lade ROT
          . . . . . . . . .
1244  C3  34  12               JP zurück          ; Rücksprung
```

Der Adreßteil wird dabei als 34 12 angegeben, da beim Z80 bei allen 16-Bit-Adressen zuerst die niederwertige Hälfte im Operationscode angegeben wird und dann die höherwertige.

Unser Programm hat noch einen kleinen Schönheitsfehler. Es arbeitet einwandfrei, wenn der Z80 mit einem Einzeltakt betrieben wird. Doch normalerweise arbeitet die CPU mit einem 2 MHz-Takt und die Ampelsequenz wird dadurch so schnell durchlaufen, daß man sie nicht mehr sieht. Nach jeder Ausgabe müßte also eine kurze Pause erfolgen, bevor der nächste Schritt erfolgen kann. Dies läßt sich mit einer sogenannten Warteschleife erreichen. Dies ist ein Programmteil, in dem die CPU sehr lange bleibt, bevor sie diesen wieder verläßt. Eine Warteschleife läßt sich zum Beispiel durch einen Zähler verwirklichen, der herunter gezählt wird. Der Zähler wird mit einer sehr großen Zahl vorbesetzt, daß selbst der Z80 eine Weile (Sekundenbereich bei unserer Ampel) braucht, um ihn herunterzuzählen.

Wie läßt sich ein Zähler realisieren? Indem der Wert 1 von einer Zelle subtrahiert wird. Und dies geschieht solange, bis der Wert = 0 ist.

Als Rückwärtszähler dient die Zelle 1300h. Sie muß am Anfang mit dem Startwert geladen werden:

```
          LD A,255d         ; laden mit maximalem Wert
          LD (1300h),A      ; und abspeichern in die Zelle
warte:    LD A,(1300h)      ; Zählschleife
          SUB 1             ; -1 bilden
          LD (1300h),A      ; wieder zurück
          JP NZ,warte       ; bis Akku = 0 wiederholen
  ---     Ende der Warteschleife ---
```

Wie lange bleibt aber der Prozessor in dieser Schleife? Dazu müssen wir die Ausführungszeit pro Befehl kennen.

LD A,(adresse) benötigt 13 Taktzyklen also 13 * 500 ns = 6.5 µs bei einem 2 MHZ Takt
SUB data benötigt 7 Taktzyklen also 7 * 500 ns = 2.3 µs
LD (adresse),A benötigt 13 Taktzyklen also 13 * 500 ns = 6.5 µs
JP NZ,adresse benötigt 10 Taktzyklen also 10 * 500 ns = 5 µs

```
                                        ; Warteschleife fuer eine Sekunde start
                                        ; auf adresse 1200h

1200    3E C8                    ld.a,200          ;aeussere Schleife 200 Mal
1202    32 1300                  ld (1300h),a      ;zaehler 1
1205    3E FF           warte1:  ld a,255          ;innere Schleife 255 Mal
1207    32 1301                  ld (1301h),a      ;zaehler 2
120A    3A 1301         warte2:  ld a,(1301h)      ;zaehler 2 innen
120D    D6 01                    sub 1
120F    32 1301                  ld (1301h),a      ;-1
1212    C2 120A                  jp nz,warte2      ;bis = 0
1215    3A 1300                  ld a,(1300h)      ;aussen zaehler 1
1218    D6 01                    sub 1
121A    32 1300                  ld (1300h),a
121D    C2 1205                  jp nz,warte1      ;innen zaehler neu belegen

                                ; --------  schleife fertig  --------
```

Abb. 8.1.3 Warteschleife

Damit ergibt sich für einen Schleifendurchlauf:

$$Tges = 6.5\ \mu s + 2.3\ \mu s + 6.5\ \mu s + 5\ \mu s = 20.3\ \mu s$$

Die Schleife wird 255mal (nicht 256) durchlaufen, also ist

$$Twarte = 255 * 20.3\ \mu s = 5176.5\ \mu s = 5.1765\ ms$$

5 ms sind aber noch zu wenig. Wir wollen z. B. 1 Sekunde haben. Dazu muß die Schleife 200mal ausgeführt werden. Wir bauen also eine Schleife um die Schleife. Das Programm zeigt *Abb. 8.1.3*. Diesmal wurde erstmalig ein automatischer Übersetzer verwendet, um das Programm zu codieren, es läßt sich aber hier noch genausogut per Hand durchführen.

Der Assembler gibt beim Objekt-Code, so nennt man die codierten Befehle, alle 16-Bit-Adressen in der lesbaren Form, also zuerst MSB und dann LBS, aus. Bei der Eingabe in den Computer muß die Reihenfolge umgekehrt werden. Beispielsweise würde im Speicher ab Adresse 1200h folgende Sequenz stehen:

```
3E C8 32 00 13 3E FF 32 01 13 3A 01 13 D6 01 32 01 13
C2 0A 12 3A 00 13 D6 01 32 00 13 C2 05 12
```

Die Warteschleife müßte nun nach jeder Ausgabe auf den IO-Port (Befehl OUT (30h),A) eingeschoben werden. Sie wird daher viermal benötigt. Das Programm wird dadurch schon recht umfangreich. Bei Computern gibt es eine Möglichkeit, Routinen, die oft benötigt werden, nur einmal abzuspeichern und von verschiedenen Programmteilen her aufrufen zu können. Das ist die Unterprogrammtechnik. In *Abb. 8.1.4* ist ein Beispiel mit einem Unterprogramm gezeigt. Das Unterprogramm UPR wird zweimal vom Hauptprogramm aus aufgerufen. Nach jedem Aufruf muß das Hauptprogramm an der Stelle weiter fortgeführt werden, an der der Aufruf erfolgt. Dazu muß sich der Prozessor die Adresse merken, bei der der Aufruf erfolgte. Dies geschieht beim Z80 mit Hilfe des Stackpointers, der im Register SP liegt. Dort ist eine Adresse (16 Bit) untergebracht, die auf eine Ramzelle zeigt. Bei einem Unterprogrammaufruf wird in zwei aufeinanderfolgende Speicherzellen, beginnend bei der Adresse des Stackpointers-1, der höherwertige Teil des

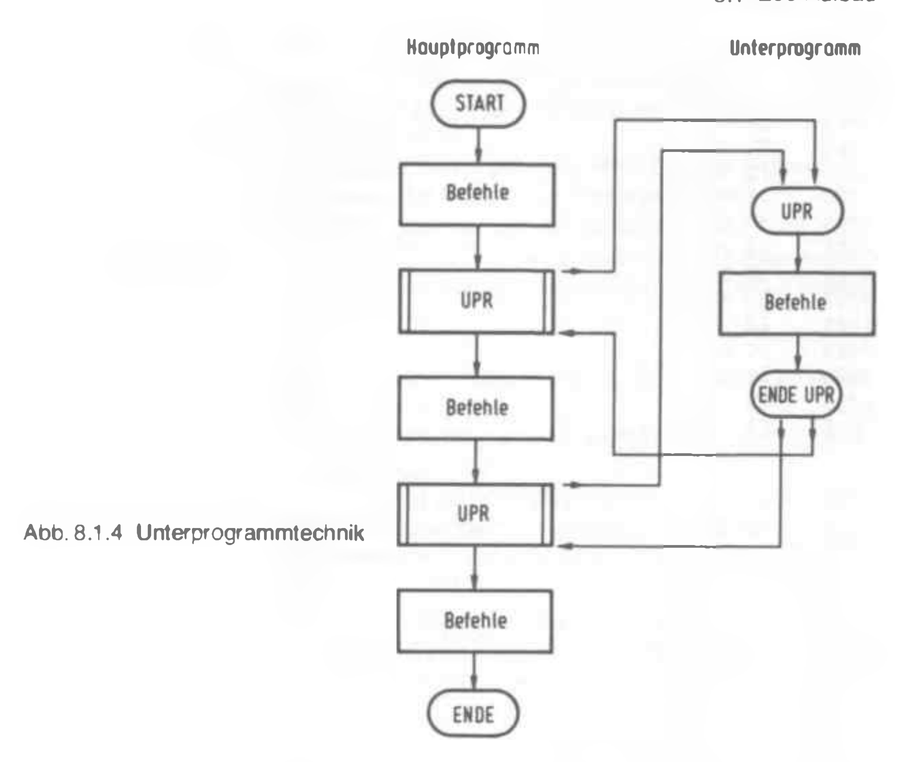


Abb. 8.1.4 Unterprogrammtechnik

Programmzählers abgespeichert und an der Stelle Stackpointer-2 wird der niederwertige Teil abgelegt. Danach wird der Stackpointer SP um zwei verringert. Der Programmzähler steht bei diesem Vorgang auf dem nächsten Befehl nach dem Aufruf-Befehl. Für das Unterprogramm gibt es einen Rücksprungbefehl. Er holt den niederwertigen Teil des Programmzählers von der Stelle SP, den höherwertigen Teil von SP+1 und anschließend wird der Stackpointer SP um zwei erhöht. Damit ist der Befehl genau die Umkehrung des Aufrufbefehls. Nun die Mnemonics der Unterprogrammbefehle:

CALL adresse   CALL 1250h   CD 50 12

Aufruf eines Unterprogramms auf Adresse 1250h. Der Programmzählerstand wird auf den Stack (so wird diese Art der Speicherung bezeichnet) gerettet. Der Stackpointer wird anschließend um zwei verringert.

RET   RET   C9

Rückkehr vom Unterprogramm in das Hauptprogramm. Die Rückkehradresse wird vom Stack zurückgeholt und der Stackpointer anschließend um zwei erhöht.

Wozu ist der Stackpointer gut, wenn es auch möglich wäre, die Rückkehradresse einfach in einem Register festzuhalten? Der Vorteil liegt darin, daß im Unterprogramm ein weiterer Unterprogrammaufruf erfolgen kann und in diesem wieder usw. Der Stackpointer wird dabei immer weiter verringert. Dies geht solange gut, bis der Speicherplatz für den Stack ausgeht, also der Bereich, den der Stackpointer mit seinen Adressen überstreicht. Es gibt auch Prozessoren, die die Rückkehradresse in ein Register legen, z. B. SCMP oder TMS 9900.

```
                                ; Unterprogrammtechnik

1200   31 13FF                          ld sp,13FFh      ;Stack pointer ans Ende
1203   3E 06                    ampel:  ld a,00000110b   ;ROT
1205   D3 30                            out (30h),a
1207   CD 1222                          call warte
120A   3E 04                            ld a,00000100b   ;ROTGELB
120C   D3 30                            out (30h),a
120E   CD 1222                          call warte
1211   3E 03                            ld a,00000011b   ;GRUEN
1213   D3 30                            out (30h),a
1215   CD 1222                          call warte
1218   3E 05                            ld a,00000101b   ;GELB
121A   D3 30                            out (30h),a
121C   CD 1222                          call warte
121F   C3 1203                          jp ampel         ;dauernd wiederholen
                                ;
                                ; unterprogramm folgt nun
                                ;
1222   3E C8                    warte:  ld a,200         ;aeussere Schleife 200 Mal
1224   32 1300                          ld (1300h),a     ;zaehler 1
1227   3E FF                    war1:   ld a,255         ;innere Schleife 255 Mal
1229   32 1301                          ld (1301h),a     ;zaehler 2
122C   3A 1301                  war2:   ld a,(1301h)     ;zaehler 2 innen
122F   D6 01                            sub 1
1231   32 1301                          ld (1301h),a     ;-1
1234   C2 122C                          jp nz,war2       ;bis = 0
1237   3A 1300                          ld a,(1300h)     ;aussen zaehler 1
123A   D6 01                            sub 1
123C   32 1300                          ld (1300h),a
123F   C2 1227                          jp nz,war1       ;innen zaehler neu belegen
1242   C9                               ret              ;Ruecksprung ins Hauptprogra
                                ;
```

Abb. 8.1.5 Beispiel eines Unterprogramms

Zurück zu unserem Verkehrsampelproblem. *Abb. 8.1.5* zeigt die Lösung mit den Unterprogrammbefehlen. Dieses Programm kann auf unserem Mikrorechner auch tatsächlich ausgeführt werden.

Mit den bisherigen Befehlen lassen sich praktisch alle programmtechnischen Aufgaben lösen. Es ist sozusagen der Grundwortschatz, mit dem man sich verständlich machen kann. Der Z80 besitzt weitere Befehle, die es erlauben, die Programme eleganter zu schreiben. So ist es möglich, anstelle des Befehls ADD A,1 den Befehl INC A zu verwenden, der den Inhalt des Akkus um eins erhöht und nur ein Byte im Speicher für den Operationscode belegt (3Ch). Auch besitzt der Z80 Möglichkeiten, um direkt Arithmetik mit 16-Bit-Daten durchzuführen, was sehr wichtig für Adreßarithmetik ist. Ferner gibt es Befehle, mit denen eine Arithmetik für beliebig lange Zahlen gemacht werden kann. Der Z80 besitzt zur Abfrage von bedingten Anweisungen, wie auch der Befehl „JP NZ,adresse" eine war, ein Register mit dem Namen Statusregister. Es hat folgende Aufteilung:

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

Die Bedeutung der einzelnen Bits:

C Carry-Flag. Es ist der Übertrag bei einer arithmetischen Operation, kann aber auch mit Schiebebefehlen verändert werden. War ein Übertrag vorhanden, so wird es gesetzt. Der Übertrag ist der der vorzeichenlosen Arithmetik.

N Additions/Subtraktions-Flag. Es hat die Aufgabe einer ggf. nachfolgenden Dezimalkorrektur zu sagen, ob zuvor eine Addition oder Subtraktion ausgeführt wurde.

P/V Paritäts- oder Überlauf-Flag. Bei logischen Operationen gibt es die Parität des Akku A an. Ist das Bit Eins, so liegt eine gerade (even) Parität vor. Bei arithmetischen Operationen ist es das Überlaufbit bei Zweierkomplement-Arithmetik (vorzeichenbehaftet). Bei Überlauf ist das Bit gesetzt.

X Ist nicht verwendet.

H Halb-Überlauf. Wird für die Dezimalkorrektur benötigt und ist der Überlauf zwischen 3tem und 4tem Bit bei Arithmetik-Operationen.

Z Zero-Flag oder Null-Flag. Es wird bei arithmetischen und logischen Operationen gesetzt und ist Eins, wenn die bearbeitete Zelle 0 ist.

S Sign-Flag. Ist das Vorzeichenbit und wird bei arithmetischen und logischen Operationen gesetzt. Dabei wird das Bit auf eins gesetzt, wenn Bit 7 der bearbeiteten Zelle eins war, entspricht also bei einer Zweier-Komplement-Arithmetik exakt dem Vorzeichen.

Die einzelnen Bits werden nicht immer alle bei allen Operationen gesetzt. Als Faustregel gilt die bei den Flags angegebene Regel. Ein paar wichtige Fälle werden wir mit den Befehlen noch kennenlernen. Die einzelnen Befehlsgruppen werden im folgenden besprochen.

1. Transportbefehle

Wir haben davon schon ein paar Befehle kennengelernt, z. B. LD A,(adresse).

Der Z80 besitzt aber eine ganze Reihe von solchen Befehlen, da er noch andere Register als A und B besitzt. *Abb. 8.1.6a . . . c* zeigt alle möglichen Kombinationen des Befehls ‚LD'. Der Z80 hat die Möglichkeit, auch mit 16-Bit-Größen zu arbeiten. Dazu können die Register BC, DE, HL sowie IX, IY und natürlich SP verwendet werden. Diese Register können direkt mit 16-Bit-Konstanten geladen werden, z. B. lädt der Befehl LD HL,1234h das Register H mit 12h und das Register L mit 34h. HL ist ein besonderes Registerpaar, es kann für eine 16-Bit-Arithmetik verwendet werden. Ebenfalls IX und IY. Diese drei Register sind dann praktisch ein neuer Akkumulator. Der Inhalt eines der anderen Registerpaare, also BC, DE und SP sowie auch HL (IX, IY) können dazu addiert werden. Nicht möglich ist es HL auf IX, IY oder umgekehrt zu addieren.

```
                ; Transport - Lade und
                ; Speicherbefehle
02                      LD      (BC),A
12                      LD      (DE),A
77                      LD      (HL),A
70                      LD      (HL),B
71                      LD      (HL),C
72                      LD      (HL),D
73                      LD      (HL),E
74                      LD      (HL),H
75                      LD      (HL),L
```

zu Abb. 8.1.6

```
36 20                   LD      (HL),' '          ;20H
DD 77 05                LD      (IX+05),A
DD 70 05                LD      (IX+05),B
DD 71 05                LD      (IX+05),C
DD 72 05                LD      (IX+05),D
DD 73 05                LD      (IX+05),E
DD 74 05                LD      (IX+05),H
DD 75 05                LD      (IX+05),L
DD 36 05 20             LD      (IX+05),20H
FD 77 05                LD      (IY+05),A
FD 70 05                LD      (IY+05),B
FD 71 05                LD      (IY+05),C
FD 72 05                LD      (IY+05),D
FD 73 05                LD      (IY+05),E
FD 74 05                LD      (IY+05),H
FD 75 05                LD      (IY+05),L
FD 36 05 20             LD      (IY+05),20H
32 84 05                LD      (0584H),A
ED 43 84 05             LD      (0584H),BC
ED 53 84 05             LD      (0584H),DE
22 84 05                LD      (0584H),HL
DD 22 84 05             LD      (0584H),IX
FD 22 84 05             LD      (0584H),IY
ED 73 84 05             LD      (0584H),SP
0A                      LD      A,(BC)
1A                      LD      A,(DE)
7E                      LD      A,(HL)
DD 7E 05                LD      A,(IX+05)
FD 7E 05                LD      A,(IY+05)
3A 84 05                LD      A,(0584H)
7F                      LD      A,A
78                      LD      A,B
79                      LD      A,C
7A                      LD      A,D
7B                      LD      A,E
7C                      LD      A,H
ED 57                   LD      A,I
7D                      LD      A,L
3E 20                   LD      A,' '   ;20H
ED 5F                   LD      A,R
46                      LD      B,(HL)
DD 46 05                LD      B,(IX+05)
FD 46 05                LD      B,(IY+05)
47                      LD      B,A
40                      LD      B,B
41                      LD      B,C
42                      LD      B,D
43                      LD      B,E
44                      LD      B,H
45                      LD      B,L
06 20                   LD      B,' '   ;20H
ED 4B 84 05             LD      BC,(0584H)
```

zu Abb. 8.1.6

```
01 84 05        LD   BC,0584H
4E              LD   C,(HL)
DD 4E 05        LD   C,(IX+05)
FD 4E 05        LD   C,(IY+05)
4F              LD   C,A
48              LD   C,B
49              LD   C,C
4A              LD   C,D
4B              LD   C,E
4C              LD   C,H
4D              LD   C,L
0E 20           LD   C,' '    ;20H
56              LD   D,(HL)
DD 56 05        LD   D,(IX+05)
FD 56 05        LD   D,(IY+05)
57              LD   D,A
50              LD   D,B
51              LD   D,C
52              LD   D,D
53              LD   D,E
54              LD   D,H
55              LD   D,L
16 20           LD   D,' '    ;20H
ED 5B 84 05     LD   DE,(0584H)
11 84 05        LD   DE,0584H
5E              LD   E,(HL)
DD 5E 05        LD   E,(IX+05)
FD 5E 05        LD   E,(IY+05)
5F              LD   E,A
58              LD   E,B
59              LD   E,C
5A              LD   E,D
5B              LD   E,E
5C              LD   E,H
5D              LD   E,L
1E 20           LD   E,' '    ;20H
66              LD   H,(HL)
DD 66 05        LD   H,(IX+05)
FD 66 05        LD   H,(IY+05)
67              LD   H,A
60              LD   H,B
61              LD   H,C
62              LD   H,D
63              LD   H,E
64              LD   H,H
65              LD   H,L
26 20           LD   H,' '    ;20H
2A 84 05        LD   HL,(0584H)
21 84 05        LD   HL,0584H
ED 47           LD   I,A
DD 2A 84 05     LD   IX,(0584H)
DD 21 84 05     LD   IX,0584H
```

zu Abb. 8.1.6

```
FD 2A 84 05        LD    IY,(0584H)
FD 21 84 05        LD    IY,0584H
6E                 LD    L,(HL)
DD 6E 05           LD    L,(IX+05)
FD 6E 05           LD    L,(IY+05)
6F                 LD    L,A
68                 LD    L,B
69                 LD    L,C
6A                 LD    L,D
6B                 LD    L,E
6C                 LD    L,H
6D                 LD    L,L
2E 20              LD    L,' '    ;20H
ED 4F              LD    R,A
ED 7B 84 05        LD    SP,(0584H)
F9                 LD    SP,HL
DD F9              LD    SP,IX
FD F9              LD    SP,IY
31 84 05           LD    SP,0584H
```

Abb. 8.1.6
Transportbefehle

Eine andere Form der Adressierung ist die sogenannte indirekte Adresse. Dazu steht z. B. im Registerpaar HL eine Adresse. Mit der Speicherzelle, die durch diese Adresse bestimmt ist, kann dann gearbeitet werden. Beispiel:

In HL steht 1350h
In der Zelle 1350h steht der Wert 55h

Mit dem Befehl LD B,(HL) wird in das Register B der Wert 55h geladen.

Die Registerpaare IX und IY haben noch eine weitere Besonderheit. Zusätzlich kann ein Verschiebefaktor, auch displacement genannt, angegeben werden. Es wird der Adresse, die im Registerpaar steht, aufaddiert und zuvor zu einer 16-Bit-Adresse ergänzt. Beispiel:

In IX steht der Wert 1310h
Mit LD C,(IX + 40h)

wird der Inhalt der Speicherzelle 1350h in das Register C geladen.

Mit LD C,(IX + 0FFh)

wird der Inhalt der Speicherzelle 130Fh in das Register C geladen (hier: 1310h + FFh → 130Fh).

Alle Transportbefehle verändern die Flags nicht. Damit ist es möglich, nach einem Transport noch den Zustand der vorherigen Operation, die die Flags verändert hat, zu sehen und eine bedingte Operation kann durchgeführt werden.

2. Austausch-Operationen

In *Abb. 8.1.7* sind Austausch-Befehle abgebildet. Sie sind eigentlich auch Transportbefehle, transportieren aber zwei oder mehr Operanden gleichzeitig. Der Z80 besitzt einen zweiten

```
                         ; Vertauschen eines Registerpaars
                         ; mit dem Inhalt des Stacks
E3                               EX      (SP),HL
DD E3                            EX      (SP),IX
FD E3                            EX      (SP),IY
                         ; Vertauschen der beiden Akkumulatoren
                         ; A und A', sowie der Flags
08                               EX      AF,AF'
                         ; Vertauschen von DE mit HL
EB                               EX      DE,HL
                         ; Vertauschen der beiden Registersaetze
                         ; BC,DE,HL
D9                               EXX
```

Abb. 8.1.7 Vertauschen von Registern

Registersatz. Die Registernamen werden mit einem Apostroph gekennzeichnet, der dem Registernamen hinten angestellt wird. Auf diese Register kann nicht direkt zugegriffen werden. Der Registersatz muß erst aktiviert werden. Dazu gibt es zwei Befehle. Der erste Befehl EX AF,AF' vertauscht die beiden Akkumulatoren und die Statusregister. Danach wird immer mit dem zweiten Satz gearbeitet. Mit dem Befehl EXX werden die Register BC', DE', HL' gegen BC, DE und HL vertauscht.

3. Blocktransportbefehle

Abb. 8.1.8 zeigt die vier Befehle. Sie sind wohl die mächtigsten Operationen, die der Z80 besitzt. Mit ihnen kann ein ganzer Datenbereich verschoben werden. Beispiel:

```
LD HL,1000h
LD DE,2000h
LD BC,500h
LDIR
```

Der Bereich 1000h bis 14FFh wird nach 2000h bis 24FFh transportiert. Mit diesem Befehl läßt sich auch das Löschen von Speicherbereichen durchführen. Beispiel:

```
LD HL,1000h
LD (HL),0
LD DE,1001h
LD BC,1FEh
LDIR
```

Der Bereich 1000h bis 11FFH wird mit dem Wert 0 belegt.

4. Stackoperationen

Der Stack, auch Keller genannt, ist vielleicht manchen Taschenrechner-Besitzern bekannt, und zwar denjenigen, die einen UPN-Rechner besitzen. Ein Stack ist ein Speicher, bei dem immer nur

```
                              ; Transportiere den Inhalt der Zelle (HL) nach
                              ; Zelle (DE)
                              ; DE,HL,BC werden anschliessend decrementiert
ED A8                               LDD
                              ; Transportiere den Inhalt der Zelle (HL) nach
                              ; Zelle (DE)
                              ; DE,HL,BC werden anschliessend decrementiert
                              ; Wiederholung bis BC=0 ist
ED B8                               LDDR
                              ; Transportiere den Inhalt der Zelle (HL) nach
                              ; Zelle (DE)
                              ; DE,HL,werden incrementiert
                              ; BC wird anschliessend decrementiert
ED A0                               LDI
                              ; Transportiere den Inhalt der Zelle (HL) nach
                              ; Zelle (DE)
                              ; DE,HL,werden incrementiert
                              ; BC wird anschliessend decrementiert
                              ; Wiederholung bis BC=0 ist
ED B0                               LDIR
```

Abb. 8.1.8 Blocktransport-Befehle

auf den zuletzt eingespeicherten Wert zugegriffen werden kann. Beim Z80 wird ein Stack mit Hilfe des Registerpaars SP aufgebaut. Dieses Registerpaar enthält dazu eine Adresse. Mit den Befehlen in *Abb. 8.1.9* kann auf den Stack zugegriffen werden. Der Stackpointer SP wird dabei automatisch verändert. Auch die Unterprogrammbefehle verwenden den Stack.

Als Beispiel soll das Registerpaar HL für eine Berechnung zwischengespeichert werden. Dies könnte mit dem Befehl LD (adresse),HL geschehen. Kürzer und eleganter geschieht es aber mit den Stackbefehlen

PUSH HL
. . diverse Operationen
POP HL

Mit dem letzten Befehl wird der alte Inhalt des Registerpaares HL wieder hergestellt. Wichtig ist bei den Befehlen, daß dazu immer zwei Operationen gehören: PUSH und POP. Danach hat der Stackpointer wieder den alten Wert. Die Operationen können aber beliebig geschachtelt werden. Nach dem PUSH-Befehl wird der Stackpointer um zwei verringert, nach dem POP-Befehl um zwei erhöht. Beispiel:

SP enthält den Wert 1234h
HL enthält den Wert 55AAh
PUSH HL

Danach enthält die Zelle 1233h den Wert 55h, also den MSB-Wert und die Zelle 1232h den Wert AAh, also LSB. Der Stackpointer steht dann auf 1232h.

Bei POP wird genau umgekehrt verfahren. Mit dem Befehl PUSH AF wird der Inhalt des Akkumulators A und das Statusregister auf den Stack geladen.

```
                        ; Das angegebene Registerpaar
                        ; wird mit dem oberen Stackwert
                        ; geladen
F1                              POP     AF
C1                              POP     BC
D1                              POP     DE
E1                              POP     HL
DD E1                           POP     IX
FD E1                           POP     IY
                        ; Der Inhalt des angegebenen Registerpaars
                        ; wird auf den Stack gespeichert
F5                              PUSH    AF
C5                              PUSH    BC
D5                              PUSH    DE
E5                              PUSH    HL
DD E5                           PUSH    IX
FD E5                           PUSH    IY
```

Abb. 8.1.9 Stack-Operationen

5. Vergleichsbefehle

Nun kommt eine sehr wichtige Gruppe *(Abb. 8.1.10)*. Mit diesen Befehlen ist es möglich, Vergleiche von Operanden durchzuführen. Dabei werden auch die Flags gesetzt. Der Befehl CP wirkt dabei wie eine Subtraktion. Das Ergebnis der Subtraktion wird allerdings nicht abgespeichert, sondern nur die Flags. Verändert werden alle Flags. Beispiele:

CP 20h

Ist im Akku A der Wert 20h enthalten, so wird das Zero-Flag gesetzt, ansonsten wird es rückgesetzt. Ist der Wert des Akkus größer oder gleich dem Wert 20h, so wird das Carry-Flag zurückgesetzt, sonst wird es gesetzt, da bei der Subtraktion „a-20h“ ein Übertrag stattfindet. Das Vorzeichen-Bit wird gemäß des Ergebnisses gesetzt.

Eine Besonderheit bieten die Blockvergleichsbefehle. Das Vorzeichen-Flag wird nach dem Ergebnis gesetzt. Das Null-Flag wird gesetzt, wenn A = (HL) gilt. Das P/V-Flag wird hier besonders verwendet. Es wird gesetzt, wenn gilt BC-1 < > 0, der Inhalt von BC vermindert um eins, ist ungleich Null. Das Carry-Flag wird nicht beeinflußt. Mit den Befehlen können Zeichen in einem Speicherbereich gesucht werden.

6. Arithmetikbefehle

Abb. 8.1.11 zeigt die Additionsbefehle. 16-Bit-Arithmetik ist ebenfalls möglich. Zu unterscheiden ist zwischen einer Addition mit Carry und einer ohne. Die Flags werden in beiden Fällen entsprechend dem Ergebnis gesetzt, auch das Carry-Flag. Aber bei der Addition mit Carry wird der Zustand des alten Carry-Flags noch hinzuaddiert. Damit läßt sich eine Mehrfach-Genaue-Arithmetik realisieren. Beispielsweise sollen eine 24-Bit-Zahl zu einer anderen 24-Bit-Zahl addiert werden. In Register BCD steht die erste Zahl, in Register EHL die zweite. Die

```
                        ; Vergleich eines Operanden mit dem
                        ; Inhalt des Akku A
BE                           CP      (HL)
DD BE 05                     CP      (IX+05)
FD 8E 05                     CP      (IY+05)
BF                           CP      A
B8                           CP      B
B9                           CP      C
BA                           CP      D
BB                           CP      E
BC                           CP      H
BD                           CP      L
FE 20                        CP      ' '      ;20H

                        ; Vergleich der Speicherstelle (HL) und
                        ; des Akku A, HL und BC wird decrementiert
ED A9                        CPD
                        ; Vergleich (HL) mit Akku A; HL,BC decrement
                        ; bis BC=0
ED B9                        CPDR
                        ; Vergleich (HL) mit Akku A; HL increment
                        ; BC decrement
ED A1                        CPI
                        ; Vergleich (HL) mit Akku A; HL increment
                        ; BC decrement bis BC=0
ED B1                        CPIR
```

Abb. 8.1.10 Vergleichsbefehle

```
                        ; Addition mit Carry
                        ; nach Akku A
89                           ADC     A,C
8A                           ADC     A,D
8B                           ADC     A,E
8C                           ADC     A,H
8D                           ADC     A,L
CE 20                        ADC     A,' '    ;20H
                        ; Addition mit Carry
                        ; nach Registerpaar HL
ED 4A                        ADC     HL,BC
ED 5A                        ADC     HL,DE
ED 6A                        ADC     HL,HL
ED 7A                        ADC     HL,SP
                        ; Addition ohne Carry
                        ; nach Akku A
81                           ADD     A,C
82                           ADD     A,D
83                           ADD     A,E
84                           ADD     A,H
85                           ADD     A,L
C6 20                        ADD     A,' '    ;20H
```

zu Abb. 8.1.11

```
                        ; Addition ohne Carry
                        ; nach Registerpaar HL
09                              ADD     HL,BC
19                              ADD     HL,DE
29                              ADD     HL,HL
39                              ADD     HL,SP
                        ; Addition ohne Carry
                        ; nach Registerpaar IX
DD 09                           ADD     IX,BC
DD 19                           ADD     IX,DE
DD 29                           ADD     IX,IX
DD 39                           ADD     IX,SP
                        ; Addition ohne Carry
                        ; nach Registerpaar IY
FD 09                           ADD     IY,BC
FD 19                           ADD     IY,DE
FD 29                           ADD     IY,IY
FD 39                           ADD     IY,SP
```

Abb. 8.1.11 Additionsbefehle

höherwertige Ziffer steht dabei in B und bei der zweiten Zahl in E. Das Ergebnis soll in EHL stehen. Es gilt dann:

```
LD A,D          ; LSB zuerst
ADD A,L         ; ohne Carry addieren
LD L,A          ; Ergebnis LSB
LD A,C
ADC A,H         ; mit Carry
LD H,A
LD A,B
ADC A,E         ; MSB mit Carry
LD E,A          ; Ergebnis in EHL
```

Bei einer 16-Bit-Addition ist das einfacher. Beispiel: Es soll 1234h zum Inhalt von HL addiert werden

```
LD BC,1234h     ; z. B. mit BC durchführen
ADD HL,BC       ; ohne Carry Ergebnis in HL
```

Bei der Subtraktion ist das genauso. *Abb. 8.1.12* zeigt die Befehle. Die Flags werden auch entsprechend gesetzt.

Ein Sonderfall ist in *Abb. 8.1.13* dargestellt. Es sind die Increment- und Decrement-Befehle. Sie können Register oder auch Speicherplätze um den Wert 1 erhöhen oder erniedrigen und eignen sich daher besonders zur Bearbeitung von Zählern. Die Flags werden aber hier besonders behandelt. Alle Doppelregister-Befehle wie z. B. INC HL verändern die Flags überhaupt nicht, bei den Einzelregister-Befehlen werden nur Zero-Flag, Sign-Flag, N-Flag und H-Flag, nicht aber das Carry-Flag beeinflußt.

```
                        ; Der angegebene Operand wird
                        ; vom Inhalt des Akkumulators
                        ; subtrahiert
96                             SUB     (HL)
DD 96 05                       SUB     (IX+05)
FD 96 05                       SUB     (IY+05)
97                             SUB     A
90                             SUB     B
91                             SUB     C
92                             SUB     D
93                             SUB     E
94                             SUB     H
95                             SUB     L
D6 20                          SUB     ' '       ;20H
                        ; Subtrahiere Operanden
                        ; von Akkumulator mit
                        ; Carry
9E                             SBC     A,(HL)
DD 9E 05                       SBC     A,(IX+05)
FD 9E 05                       SBC     A,(IY+05)
9F                             SBC     A,A
98                             SBC     A,B
99                             SBC     A,C
9A                             SBC     A,D
9B                             SBC     A,E
9C                             SBC     A,H
9D                             SBC     A,L
DE 20                          SBC     A,' '     ;20H
                        ; Subtrahiere Operanden
                        ; von Registerpaar HL
                        ; mit Carry
ED 42                          SBC     HL,BC
ED 52                          SBC     HL,DE
ED 62                          SBC     HL,HL
ED 72                          SBC     HL,SP
```

Abb. 8.1.12 Subtraktionsbefehle

```
                        ; Incrementieren eines Operanden
                        ; Der Wert des Operanden wird um
                        ; Eins erhoeht. Bei Registerpaarangabe
                        ; wird mit 16 Bits gearbeitet
34                             INC     (HL)
DD 34 05                       INC     (IX+05)
FD 34 05                       INC     (IY+05)
3C                             INC     A
04                             INC     B
03                             INC     BC
0C                             INC     C
14                             INC     D
13                             INC     DE
```

zu Abb. 8.1.13

```
1C                      INC     E
24                      INC     H
23                      INC     HL
DD 23                   INC     IX
FD 23                   INC     IY
2C                      INC     L
33                      INC     SP
                  ; Decrementieren eines Operanden,
                  ; der Operandenwert wird um eins
                  ; verringert
                  ; Dabei werden Doppelregister als
                  ; 16-Bit Groessen behandelt
35                      DEC     (HL)
DD 35 05                DEC     (IX+05)
FD 35 05                DEC     (IY+05)
3D                      DEC     A
05                      DEC     B
0B                      DEC     BC
0D                      DEC     C
15                      DEC     D
1B                      DEC     DE
1D                      DEC     E
25                      DEC     H
2B                      DEC     HL
DD 2B                   DEC     IX
FD 2B                   DEC     IY
2D                      DEC     L
3B                      DEC     SP
```

Abb. 8.1.13 Increment- und Decrement-Befehle

7. Logische Operationen

Neben arithmetischen Befehlen gibt es bei Computern auch Befehle, um logische Verknüpfungen durchzuführen. *Abb. 8.1.14* zeigt die Möglichkeiten. Es gibt dabei drei verschiedene Operationen, AND, OR und XOR, die den Verknüpfungen UND, ODER und EXCLUSIV-ODER entsprechen. Die Verknüpfungen werden in dem Akkumulator bit-weise durchgeführt. Beispiel:

LD A, 10110011b
AND 11011101b

Anschließend steht im Akku der Wert 10010001b.
Wird dann der Befehl

OR 11110000b

durchgeführt, so ergibt sich 11110001b im Akku. Folgt dann

XOR 11001100b

ist der Akkuinhalt schließlich 00111101b.

```
                        ; Logisch Und-Verknuepfung
                        ; nach Akku A
A6                              AND     (HL)
DD A6 05                        AND     (IX+05)
FD A6 05                        AND     (IY+05)
A7                              AND     A
A0                              AND     B
A1                              AND     C
A2                              AND     D
A3                              AND     E
A4                              AND     H
A5                              AND     L
E6 20                           AND     ' '     ;20H
                        ; Logisch Oder-Verknuepfung von Operand
                        ; und Akku A
B6                              OR      (HL)
DD B6 05                        OR      (IX+05)
FD B6 05                        OR      (IY+05)
B7                              OR      A
B0                              OR      B
B1                              OR      C
B2                              OR      D
B3                              OR      E
B4                              OR      H
B5                              OR      L
F6 20                           OR      ' '     ;20H
                        ; Logische Exclusiv-Oder-Verknuepfung
                        ; eines Operanden mit dem Akku A
AE                              XOR     (HL)
DD AE 05                        XOR     (IX+05)
FD AE 05                        XOR     (IY+05)
AF                              XOR     A
A8                              XOR     B
A9                              XOR     C
AA                              XOR     D
AB                              XOR     E
AC                              XOR     H
AD                              XOR     L
EE 20                           XOR     ' '     ;20H
                        ;
                        ;
```

Abb. 8.1.14 Logische Verknüpfungen

8. Diverse Einzelbefehle

Abb. 8.1.15 zeigt eine Reihe einzelner Befehle. Mit dem Befehl CPL ist es zum Beispiel möglich, das Einerkomplement einer Zahl zu bilden. Beispiel:

```
Akku vorher:   11011011b
               CPL
nachher:       00100100b
```

```
                    ; Carry-Flag komplementieren
3F                        CCF
                    ; Einer-Komplement des Akkus wird
                    ; gebildet
2F                        CPL
                    ; Dezimal-Korrektur wird ausgefuehrt
27                        DAA
                    ; Sperren des Interrupts
F3                        DI
                    ; Freigabe des Interrupts
FB                        EI

                    ; Stop. Warten auf RESET oder Interrupt
76                        HALT
                    ; Interrupt Mode setzen
                    ; IM 0 nach Reset voreingestellt
ED 46                     IM     0
ED 56                     IM     1
ED 5E                     IM     2

                    ; Bilde das Zweier-Komplement des Akku A
ED 44                     NEG
                    ; Keine Operation
00                        NOP

                    ; Setze das Carry-Flag
37                        SCF
```

Abb. 8.1.15 Einzelbefehle

Mit dem Zweierkomplementbefehl sieht das anders aus:

```
Akku vorher:    11011011b
                NEG
Akku nachher    00100101b
```

Der NOP-Befehl bewirkt nichts und wird gern als Verzögerung verwendet.

Der Befehl DAA kann zur Dezimal-Korrektur verwendet werden, um mit BCD-Zahlen zu arbeiten. Beispiel:

```
Akku: 49h       ; BCD ZAHL 4 9
ADD A,27h       ; BCD ZAHL 2 7
DAA             ; BCD ZAHL 7 6
```

Die Subtraktion ist auch möglich.

9. Rotationsbefehle

Mit der Rotation kann im Prinzip ein Schieberegister realisiert werden. Dazu gibt es verschiedene Varianten, die die *Abb. 8.1.16* zeigt. Dort sind logische Operationen enthalten, die das Register komplett, also im Kreis herum rotieren lassen. Die Befehle, die durch das Carry hindurchschieben, schieben eigentlich 9 Bits.

Beispiele	Akku zuvor auf 10011001b	Carry = 0
RLCA		
Akku	00110011b	Carry = 1
RLA		
Akku	01100111b	Carry = 0

bei RRCA und RRA entsprechend umgekehrt.

Eine Besonderheit sind die Befehle RLD und RRD. Dort können Operationen auf Digits durchgeführt werden, also mit 4-Bit-Größen. Bei RLD werden die Bits 0 . . 3 des Akkus nach (HL) Bits 0 . . 3 transportiert, die alten Bits 0 . . 3 von (HL) (sprich Speicherzelle, die durch HL adressiert wurde), werden nach Bits 4 . . 7 von (HL) transportiert und diese alten Bits nach Bits 0 . . 3 des Akkus. Der Befehl RRD kehrt den Vorgang exakt um.

```
                        ; Rotiere Links durch das Carry-Flag
CB 16                          RL      (HL)
DD CB 05 16                    RL      (IX+05)
FD CB 05 16                    RL      (IY+05)
CB 17                          RL      A
CB 10                          RL      B
CB 11                          RL      C
CB 12                          RL      D
CB 13                          RL      E
CB 14                          RL      H
CB 15                          RL      L
                        ; Rotiere den Inhalt des Akkumulators
                        ; links durch das Carry-Flag
17                             RLA
                        ; Rotiere Links circular
CB 06                          RLC     (HL)
DD CB 05 06                    RLC     (IX+05)
FD CB 05 06                    RLC     (IY+05)
CB 07                          RLC     A
CB 00                          RLC     B
CB 01                          RLC     C
CB 02                          RLC     D
CB 03                          RLC     E
CB 04                          RLC     H
CB 05                          RLC     L
                        ; Rotiere den Inhalt des
                        ; Akkumulators links circular
07                             RLCA
                        ; Rotiere eine Digitgroesse (4Bit)
                        ; links und rechts zwischen Akku A
                        ; und der Zelle (HL)
ED 6F                          RLD
```

zu Abb. 8.1.16

```
                      ; Rotiere rechts durch das Carry-Flag
CB 1E                         RR      (HL)
DD CB 05 1E                   RR      (IX+05)
FD CB 05 1E                   RR      (IY+05)
CB 1F                         RR      A
CB 18                         RR      B
CB 19                         RR      C
CB 1A                         RR      D
CB 1B                         RR      E
CB 1C                         RR      H
CB 1D                         RR      L
                      ; Rotiere den Inhalt des Akkumulators
                      ; rechst durch das Carry-Flag
1F                            RRA
                      ; Rotiere rechts circular
CB 0E                         RRC     (HL)
DD CB 05 0E                   RRC     (IX+05)
FD CB 05 0E                   RRC     (IY+05)
CB 0F                         RRC     A
CB 08                         RRC     B
CB 09                         RRC     C
CB 0A                         RRC     D
CB 0B                         RRC     E
CB 0C                         RRC     H
CB 0D                         RRC     L
```

Abb. 8.1.16 Schiebebefehle Rotationen

In *Abb. 8.1.17* sind noch weitere Schiebebefehle abgebildet. Bei den arithmetischen Schiebebefehlen wird so verfahren, als ob eine Division durch zwei bei den Rechts-Schiebebefehlen und eine Multiplikation mit 2 bei den Links-Schiebebefehlen durchgeführt wird. Das Carry-Flag wird als Überlaufbit behandelt.

```
                      ; Rotiere den Inhalt des Akku A
                      ; rechts circular
0F                            RRCA
                      ; Rotiere ein Digit (4Bits)
                      ; rechts und links zwischen
                      ; Akku A und Zelle (HL)
ED 67                         RRD
                      ; Arithmetische Schiebeoperation
                      ; nach links (nachfuellen mit 0)
CB 26                         SLA     (HL)
DD CB 05 26                   SLA     (IX+05)
FD CB 05 26                   SLA     (IY+05)
CB 27                         SLA     A
CB 20                         SLA     B
CB 21                         SLA     C
CB 22                         SLA     D
CB 23                         SLA     E
CB 24                         SLA     H
CB 25                         SLA     L
```

zu Abb. 8.1.17

```
                            ; Arithmetische Schieboperation
                            ; nach rechst (duplizieren des
                            ; Vorzeichen Bits 7)
CB 2E                               SRA     (HL)
DD CB 05 2E                         SRA     (IX+05)
FD CB 05 2E                         SRA     (IY+05)
CB 2F                               SRA     A
CB 28                               SRA     B
CB 29                               SRA     C
CB 2A                               SRA     D
CB 2B                               SRA     E
CB 2C                               SRA     H
CB 2D                               SRA     L
                            ; Logisches Schieben nach rechts
                            ; mit 0 nachfuellen
CB 3E                               SRL     (HL)
DD CB 05 3E                         SRL     (IX+05)
FD CB 05 3E                         SRL     (IY+05)
CB 3F                               SRL     A
CB 38                               SRL     B
CB 39                               SRL     C
CB 3A                               SRL     D
CB 3B                               SRL     E
CB 3C                               SRL     H
CB 3D                               SRL     L
```

Abb. 8.1.17 Schiebebefehle

10. Bit-Operationen

Sehr komfortabel sind die Bitmanipulations-Befehle des Z80. *Abb. 8.1.18* zeigt die Befehle, um den Zustand eines Bits abzufragen. Das Zero-Flag wird dabei exakt so gesetzt, wie der Zustand des Bits war. War der Zustand des Bits eine Eins, so wird das Null-Flag rückgesetzt, um anzuzeigen, daß eine NICHT-NULL-Bedingung vorliegt.

Abb. 8.1.19a + b zeigen Befehle, mit denen ein einzelnes Bit gesetzt werden kann. Beispiel:

Akku vorher   10000100b
SET 4,A
Akku nachher   10010100b

Mit den Befehlen aus *Abb. 8.1.20a + b* können einzelne Bits auch wieder zurückgesetzt werden.

11. Sprungbefehle

Abb. 8.1.21 zeigt verschiedene Sprungbefehle. Dabei hat der Z80 zwei unterschiedliche Arten. Die Sprünge mit absoluten Adressen und solche mit einer relativen Adresse. Bei den Sprüngen mit relativer Adresse ergeben sich zwei Vorteile. Zum einen ist der Operations-Code kürzer und ferner wird das Programmstück unabhängig von der Lage im Speicher. Dazu wird der angegebene

```
                        ; Testen eines Bits
                        ; setzen des Zero-Flags
                        ; entsprechend des Wertes
CB 46                           BIT     0,(HL)
DD CB 05 46                     BIT     0,(IX+05)
FD CB 05 46                     BIT     0,(IY+05)
CB 47                           BIT     0,A
CB 40                           BIT     0,B
CB 41                           BIT     0,C
CB 42                           BIT     0,D
CB 43                           BIT     0,E
CB 44                           BIT     0,H
CB 45                           BIT     0,L
CB 4E                           BIT     1,(HL)
DD CB 05 4E                     BIT     1,(IX+05)
FD CB 05 4E                     BIT     1,(IY+05)
CB 4F                           BIT     1,A
CB 48                           BIT     1,B
CB 49                           BIT     1,C
CB 4A                           BIT     1,D
CB 4B                           BIT     1,E
CB 4C                           BIT     1,H
CB 4D                           BIT     1,L
CB 56                           BIT     2,(HL)
DD CB 05 56                     BIT     2,(IX+05)
FD CB 05 56                     BIT     2,(IY+05)
CB 57                           BIT     2,A
CB 50                           BIT     2,B
CB 51                           BIT     2,C
CB 52                           BIT     2,D
CB 53                           BIT     2,E
CB 54                           BIT     2,H
CB 55                           BIT     2,L
CB 5E                           BIT     3,(HL)
DD CB 05 5E                     BIT     3,(IX+05)
FD CB 05 5E                     BIT     3,(IY+05)
CB 5F                           BIT     3,A
CB 58                           BIT     3,B
CB 59                           BIT     3,C
CB 5A                           BIT     3,D
CB 5B                           BIT     3,E
CB 5C                           BIT     3,H
CB 5D                           BIT     3,L
CB 66                           BIT     4,(HL)
DD CB 05 66                     BIT     4,(IX+05)
FD CB 05 66                     BIT     4,(IY+05)
CB 67                           BIT     4,A
CB 60                           BIT     4,B
CB 61                           BIT     4,C
CB 62                           BIT     4,D
CB 63                           BIT     4,E
CB 64                           BIT     4,H
```

zu Abb. 8.1.18

```
CB 65                          BIT      4,L
CB 6E                          BIT      5,(HL)
DD CB 05 6E                    BIT      5,(IX+05)
FD CB 05 6E                    BIT      5,(IY+05)
CB 6F                          BIT      5,A
CB 68                          BIT      5,B
CB 69                          BIT      5,C
CB 6A                          BIT      5,D
CB 6B                          BIT      5,E
CB 6C                          BIT      5,H
CB 6D                          BIT      5,L
CB 76                          BIT      6,(HL)
DD CB 05 76                    BIT      6,(IX+05)
FD CB 05 76                    BIT      6,(IY+05)
CB 77                          BIT      6,A
CB 70                          BIT      6,B
CB 71                          BIT      6,C
CB 72                          BIT      6,D
CB 73                          BIT      6,E
CB 74                          BIT      6,H
CB 75                          BIT      6,L
CB 7E                          BIT      7,(HL)
DD CB 05 7E                    BIT      7,(IX+05)
FD CB 05 7E                    BIT      7,(IY+05)
CB 7F                          BIT      7,A
CB 78                          BIT      7,B
CB 79                          BIT      7,C
CB 7A                          BIT      7,D
CB 7B                          BIT      7,E
CB 7C                          BIT      7,H
CB 7D                          BIT      7,L
```

Abb. 8.1.18 Einzelbittest-Befehle

```
                ; Setzen eines Bits
CB C6                  SET      0,(HL)
DD CB 05 C6            SET      0,(IX+05)
FD CB 05 C6            SET      0,(IY+05)
CB C7                  SET      0,A
CB C0                  SET      0,B
CB C1                  SET      0,C
CB C2                  SET      0,D
CB C3                  SET      0,E
CB C4                  SET      0,H
CB C5                  SET      0,L
CB CE                  SET      1,(HL)
DD CB 05 CE            SET      1,(IX+05)
FD CB 05 CE            SET      1,(IY+05)
CB CF                  SET      1,A
CB C8                  SET      1,B
CB C9                  SET      1,C
CB CA                  SET      1,D
```

zu Abb. 8.1.19

```
CB CB          SET   1,E
CB CC          SET   1,H
CB CD          SET   1,L
CB D6          SET   2,(HL)
DD CB 05 D6    SET   2,(IX+05)
FD CB 05 D6    SET   2,(IY+05)
CB D7          SET   2,A
CB D0          SET   2,B
CB D1          SET   2,C
CB D2          SET   2,D
CB D3          SET   2,E
CB D4          SET   2,H
CB D5          SET   2,L
CB DE          SET   3,(HL)
DD CB 05 DE    SET   3,(IX+05)
FD CB 05 DE    SET   3,(IY+05)
CB DF          SET   3,A
CB D8          SET   3,B
CB D9          SET   3,C
CB DA          SET   3,D
CB DB          SET   3,E
CB DC          SET   3,H
CB DD          SET   3,L
CB E6          SET   4,(HL)
DD CB 05 E6    SET   4,(IX+05)
FD CB 05 E6    SET   4,(IY+05)
CB E7          SET   4,A
CB E0          SET   4,B
CB E1          SET   4,C
CB E2          SET   4,D
CB E3          SET   4,E
CB E4          SET   4,H
CB E5          SET   4,L
CB EE          SET   5,(HL)
DD CB 05 EE    SET   5,(IX+05)
FD CB 05 EE    SET   5,(IY+05)
CB EF          SET   5,A
CB E8          SET   5,B
CB E9          SET   5,C
CB EA          SET   5,D
CB EB          SET   5,E
CB EC          SET   5,H
CB ED          SET   5,L
CB F6          SET   6,(HL)
DD CB 05 F6    SET   6,(IX+05)
FD CB 05 F6    SET   6,(IY+05)
CB F7          SET   6,A
CB F0          SET   6,B
CB F1          SET   6,C
CB F2          SET   6,D
CB F3          SET   6,E
CB F4          SET   6,H
```

zu Abb. 8.1.19

```
CB F5                SET     6,L
CB FE                SET     7,(HL)
DD CB 05 FE          SET     7,(IX+05)
FD CB 05 FE          SET     7,(IY+05)
CB FF                SET     7,A
CB F8                SET     7,B
CB F9                SET     7,C
CB FA                SET     7,D
CB FB                SET     7,E
CB FC                SET     7,H
CB FD                SET     7,L
```

Abb. 8.1.19 Einzelbitsetz-Befehle

```
                      ; Ruecksetzen des angegebenen Bits in
CB 86                        RES     0,(HL)
DD CB 05 86                  RES     0,(IX+05)
FD CB 05 86                  RES     0,(IY+05)
CB 87                        RES     0,A
CB 80                        RES     0,B
CB 81                        RES     0,C
CB 82                        RES     0,D
CB 83                        RES     0,E
CB 84                        RES     0,H
CB 85                        RES     0,L
CB 8E                        RES     1,(HL)
DD CB 05 8E                  RES     1,(IX+05)
FD CB 05 8E                  RES     1,(IY+05)
CB 8F                        RES     1,A
CB 88                        RES     1,B
CB 89                        RES     1,C
CB 8A                        RES     1,D
CB 8B                        RES     1,E
CB 8C                        RES     1,H
CB 8D                        RES     1,L
CB 96                        RES     2,(HL)
DD CB 05 96                  RES     2,(IX+05)
FD CB 05 96                  RES     2,(IY+05)
CB 97                        RES     2,A
CB 90                        RES     2,B
CB 91                        RES     2,C
CB 92                        RES     2,D
CB 93                        RES     2,E
CB 94                        RES     2,H
CB 95                        RES     2,L
CB 9E                        RES     3,(HL)
DD CB 05 9E                  RES     3,(IX+05)
FD CB 05 9E                  RES     3,(IY+05)
CB 9F                        RES     3,A
CB 98                        RES     3,B
CB 99                        RES     3,C
```

zu Abb. 8.1.20

```
CB 9A             RES  3,D
CB 9B             RES  3,E
CB 9C             RES  3,H
CB 9D             RES  3,L
CB A6             RES  4,(HL)
DD CB 05 A6       RES  4,(IX+05)
FD CB 05 A6       RES  4,(IY+05)
CB A7             RES  4,A
CB A0             RES  4,B
CB A1             RES  4,C
CB A2             RES  4,D
CB A3             RES  4,E
CB A4             RES  4,H
CB A5             RES  4,L
CB AE             RES  5,(HL)
DD CB 05 AE       RES  5,(IX+05)
FD CB 05 AE       RES  5,(IY+05)
CB AF             RES  5,A
CB A8             RES  5,B
CB A9             RES  5,C
CB AA             RES  5,D
CB AB             RES  5,E
CB AC             RES  5,H
CB AD             RES  5,L
CB B6             RES  6,(HL)
DD CB 05 B6       RES  6,(IX+05)
FD CB 05 B6       RES  6,(IY+05)
CB B7             RES  6,A
CB B0             RES  6,B
CB B1             RES  6,C
CB B2             RES  6,D
CB B3             RES  6,E
CB B4             RES  6,H
CB B5             RES  6,L
CB BE             RES  7,(HL)
DD CB 05 BE       RES  7,(IX+05)
FD CB 05 BE       RES  7,(IY+05)
CB BF             RES  7,A
CB B8             RES  7,B
CB B9             RES  7,C
CB BA             RES  7,D
CB BB             RES  7,E
CB BC             RES  7,H
CB BD             RES  7,L
```

Abb. 8.1.20 Einzelbitrücksetz-Befehle

```
                  ; Unbedingte Spruenge Indirekt
E9                JP   (HL)
DD E9             JP   (IX)
FD E9             JP   (IY)
```

zu Abb. 8.1.21

```
                          ; Unbedingter Sprung Absolut
C3 84 05                         JP      0584H
                          ; Bedingte Spruenge Absolut
DA 84 05                         JP      C,0584H
FA 84 05                         JP      M,0584H
D2 84 05                         JP      NC,0584H
C2 84 05                         JP      NZ,0584H
F2 84 05                         JP      P,0584H
EA 84 05                         JP      PE,0584H
E2 84 05                         JP      PO,0584H
CA 84 05                         JP      Z,0584H
                          ; Bedingte Spruenge Relativ
38 FE                     L1:    JR      C,L1
30 FE                     L2:    JR      NC,L2
20 FE                     L3:    JR      NZ,L3
28 FE                     L4:    JR      Z,L4
                          ; Unbedingter Sprung Relativ
18 FE                     L5:    JR      L5

                          ; Das Register B wird um eins
                          ; verringert, der Sprung wird
                          ; ausgefuehrt bis das Register B
                          ; den Wert 0 besitzt
10 FE                     LP:    DJNZ    LP
```

Abb. 8.1.21 Sprungbefehle

8-Bit-Wert als Zweierkomplement-Zahl aufgefaßt und zum aktuellen Stand des Programmzählers addiert. Beispiel:

```
                JR SKIP
                NOP
SKIP:           NOP
```

ist codiert:

```
18 01           JR SKIP
00              NOP
00   SKIP:      NOP
```

Bei einem Sprung zurück:

```
LOOP:           NOP
                NOP
                JR LOOP
```

oder codiert:

```
00   LOOP:      NOP
00              NOP
18 FC           JR LOOP
```

Der Befehl DJNZ ist noch eine Delikatesse. Er decrementiert das Register B und springt in Abhängigkeit des Wertes.

Damit lassen sich elegant Schleifen aufbauen. Beispiel:

```
              LD B,5
LOOP:         NOP
              NOP
              DJNZ LOOP
```

Die Schleife wird fünf Mal durchlaufen.

12. Unterprogramm-Aufrufe

Abb. 8.1.22 zeigt alle Unterprogramm-Befehle. Dabei gibt es, sowohl bei den Aufrufen, als auch bei Rücksprüngen, bedingte Anweisungen.

```
                ; Bedingter Unterprogrammaufruf
                ; falls die Bedingung erfuellt ist
DC 84 05                CALL    C,0584H
FC 84 05                CALL    M,0584H
D4 84 05                CALL    NC,0584H
C4 84 05                CALL    NZ,0584H
F4 84 05                CALL    P,0584H
EC 84 05                CALL    PE,0584H
E4 84 05                CALL    PO,0584H
CC 84 05                CALL    Z,0584H
                ; Unbedingter Unterprogrammaufruf
CD 84 05                CALL    0584H


                ; Restart auf Speicherzelle
                ;l wie Unterprogrammaufruf
C7                      RST     00H
CF                      RST     08H
D7                      RST     10H
DF                      RST     18H
E7                      RST     20H
EF                      RST     28H
F7                      RST     30H
FF                      RST     38H
                ; Ruecksprung aus einem Unterprogramm
C9                      RET
                ; Bedingter Ruecksprung aus einem
                ; Unterprogramm
D8                      RET     C
F8                      RET     M
D0                      RET     NC
```

zu Abb. 8.1.22

```
C0                          RET     NZ
F0                          RET     P
E8                          RET     PE
E0                          RET     PO
C8                          RET     Z
                      ; Rueckkehr aus einem Interruptprogramm
                      ; das mit NMI aufgerufen wurde
ED 4D                       RETI
                      ; Rueckkehr aus einem Interruptprogramm
                      ; das mit INT aufgerufen wurde
ED 45                       RETN
```

Abb. 8.1.22 Unterprogramm-Befehle

Die RST-Befehle sind ebenfalls Unterprogrammaufrufe, jedoch können sie nur auf 8 vorbestimmte Adressen ausgeführt werden und sind als Hilfsmittel bei der Interrupt-Verarbeitung verwendbar. Bei einem Interrupt wird ebenfalls nur ein Unterprogrammaufruf durchgeführt.

13. I/O-Befehle

Abb. 8.1.23 zeigt alle Standard-I/O-Befehle. Besondere Bedeutung erlangen die I/O-Blockbefehle aus *Abb. 8.1.24*. Mit diesen Befehlen ist es möglich, einen ganzen Speicherblock (maximal 256 Bytes lang) auszulagern oder zu laden.

```
                      ; Laden eines Registers von einem
                      ; IO-Geraet mit der Adresse, die
                      ; in dem Register C steht
ED 78                       IN      A,(C)
ED 40                       IN      B,(C)
ED 48                       IN      C,(C)
ED 50                       IN      D,(C)
ED 58                       IN      E,(C)
ED 60                       IN      H,(C)
ED 68                       IN      L,(C)


                      ; Laden des Akkumulators mit einem Wert
                      ; des IO-Geraets auf der angegebenen
                      ; Adresse
DB 20                       IN      A,(20H)


                      ; Ausgabe des Inhalt von Akku A an das
                      ; Port mit der Adresse die in C steht
                      ; (C).
```

zu Abb. 8.1.23

```
ED 79                  OUT     (C),A
ED 41                  OUT     (C),B
ED 49                  OUT     (C),C
ED 51                  OUT     (C),D
ED 59                  OUT     (C),E
ED 61                  OUT     (C),H
ED 69                  OUT     (C),L
```

Abb. 8.1.23 IO-Befehle

```
                    ; Der Port mit der angegebenen Adresse
                    ; wird mit dem Inhalt des Akkus geladen
D3 20                    OUT     (20H),A
```

14. Interrupt-Verarbeitung

Eine sehr wichtige Fähigkeit bei Computern ist die Interrupt-Behandlung. Ein Interrupt ist ein durch ein Hardwaresignal ausgelöster Unterprogrammaufruf. Damit kann durch ein Ereignis eine Aktion im Rechner ausgelöst werden. Gäbe es keine Interruptverarbeitung, so müßte im laufenden Programm ständig abgefragt werden, ob dieses Ereignis vorliegt oder nicht. Wenn ein solches Ereignis zu jeder Zeit kommen kann, wird das Hauptprogramm durch die dauernden Abfragen sehr langsam.

```
                    ; Laden der Speicherstelle (HL) mit einem
ED AA               ; Wert des Input-Ports mit Adresse (C)
                    ; HL, B decrementieren
                         IND
                    ; Laden der Speicherstelle (HL) mit einem
                    ; Wert des Input-Ports mit Adresse (C)
ED BA               ; HL, B decrementieren, Wiederholung
                    ; bis B=0
                         INDR
                    ; Laden der Speicherstelle (HL) mit einem
ED A2               ; Wert des Input-Ports mit Adresse (C)
                    ; HL Incrementieren, B decrementieren
                         INI
                    ; Laden der Speicherstelle (HL) mit einem
                    ; Wert des Input-Ports mit Adresse (C)
ED B2               ; HL Incrementieren, B decrementieren
                    ; Wiederholung bis B=0
                         INIR

                    ; Der Ausgabeport mit Adresse (C) wird von
                    ; der Adresse (HL) geladen
                    ; HL und B werden decrementiert
```

zu Abb. 8.1.24

```
ED AB                            OUTD
                         ; Der Ausgabeport mit Adresse (C) wird von
                         ; der Adresse (HL) geladen
                         ; HL und B werden decrementiert
                         ; Wiederholung bis B=0 ist
ED BB                            OTDR
                         ; Der Ausgabeport mit Adresse (C) wird von
                         ; der Adresse (HL) geladen
                         ; HL wird incrementiert
                         ; und B wird decrementiert
ED A3                            OUTI
                         ; Der Ausgabeport mit Adresse (C) wird von
                         ; der Adresse (HL) geladen
                         ; HL wird incrementiert
                         ; und B wird decrementiert
                         ; Wiederholung bis B=0 ist
ED B3                            OTIR
```

Abb. 8.1.24 Block-I/O-Befehle

8.1.1 Assembler

In diesem Abschnitt wird etwas über die Funktionsweise von Assemblern sowie deren Fähigkeiten gesagt.

Die Aufgabe eines Assemblers ist es, ein mit Mnemonics geschriebenes Programm in den Maschinencode zu übersetzen. Bei Sprüngen wird zum Beispiel noch ein Sprungziel angegeben. Dieses Sprungziel kann in symbolischer Form, z. B. als Name gegeben werden. Diese Namen müssen vom Assembler durch die Maschinenadresse ersetzt werden.

Wie bei der Übersetzung eines Programms vorgegangen wird, sei im folgenden, anhand des Programms aus *Abb. 8.1.25* gezeigt.

```
;************************************
;* Beispiel einer Handuebersetzung *
;************************************


start:  ld sp,13ffh     ;stack definieren
        call init       ;forwaerts referenz
        call main       ;forwaerts referenz
        jp start        ;rueckwaerts referenz

init:   ld a,5
        out (30h),a     ;port init
        ret

main:   ld b,5          ;schleifenzaehler
        ld a,1          ;ausgabewert
```

zu Abb. 8.1.25

```
loop:   out (31h),a         ;dahin ausgeben
        djnz loop           ;rueckwaertsref
        in a,(31h)
        or a                ;test ob null
        jr z,skip           ;forwaerts referenz
        ld a,6
        out (31h),a
        jp over             ;forwaerts referenz
skip:   ld a,7
        out (31h),a
over:   ld a,8
        out (31h),a
        ret
```

Abb. 8.1.25 Beispiel einer Handübersetzung

```
                              ;**********************************
                              ;* Beispiel einer Handuebersetzung *
                              ;**********************************
0000    31 FF 13              start: ld sp,13ffh       ;stack definieren
0003    CD ?? ??                     call init         ;forwaerts referenz
0006    CD ?? ??                     call main         ;forwaerts referenz
0009    C3 00 00                     jp start          ;rueckwaerts referenz

000C    3E 05                 init:  ld a,5
000E    D3 30                        out (30h),a       ;port init
0010    C9                           ret

0011    06 05                 main:  ld b,5            ;schleifenzaehler
0013    3E 01                        ld a,1            ;ausgabewert
0015    D3 31                 loop:  out (31h),a       ;dahin ausgeben
0017    10 FC                        djnz loop         ;rueckwaertsref
0019    DB 31                        in a,(31h)
001B    B7                           or a              ;test ob null
001C    28 ??                        jr z,skip         ;forwaerts referenz
001E    3E 06                        ld a,6
0020    D3 31                        out (31h),a
0022    C3 ?? ??                     jp over           ;forwaerts referenz
0025    3E 07                 skip:  ld a,7
0027    D3 31                        out (31h),a
0029    3E 08                 over:  ld a,8
002B    D3 31                        out (31h),a
002D    C9                           ret
```

Abb. 8.1.26 Pass 1

Aufgabe ist es, das Programm von Hand zu übersetzen. Dazu wird in mehreren Schritten vorgegangen. Als erstes werden alle Operationscodes erzeugt. Dabei können Sprungadressen, die noch nicht bekannt sind, auch noch nicht eingesetzt werden. Ein Listing nach *Abb. 8.1.26* entsteht dabei. Die Teile mit ?? sind noch unbekannt. Nach diesem ersten Durchlauf (engl. PASS), muß ein erneuter Durchlauf erfolgen, in dem die noch fehlenden Adreßteile eingesetzt werden. Danach ergibt sich ein Listing nach *Abb. 8.1.27*. Ein Assembler geht dabei genauso vor. Er muß den

```
                              ;***********************************
                              ;* Beispiel einer Handuebersetzung *
                              ;***********************************

0000    31 FF 13              start: ld sp,13ffh       ;stack definieren
0003    CD 0C 00                     call init         ;forwaerts referenz
0006    CD 11 00                     call main         ;forwaerts referenz
0009    C3 00 00                     jp start          ;rueckwaerts referenz

000C    3E 05                 init:  ld a,5
000E    D3 30                        out (30h),a       ;port init
0010    C9                           ret

0011    06 05                 main:  ld b,5            ;schleifenzaehler
0013    3E 01                        ld a,1            ;ausgabewert
0015    D3 31                 loop:  out (31h),a       ;dahin ausgeben
0017    10 FC                        djnz loop         ;rueckwaertsref
0019    DB 31                        in a,(31h)
001B    B7                           or a              ;test ob null
001C    28 07                        jr z,skip         ;forwaerts referenz
001E    3E 06                        ld a,6
0020    D3 31                        out (31h),a
0022    C3 29 00                     jp over           ;forwaerts referenz
0025    3E 07                 skip:  ld a,7
0027    D3 31                        out (31h),a
0029    3E 08                 over:  ld a,8
002B    D3 31                        out (31h),a
002D    C9                           ret
```

Abb. 8.1.27 Pass 2

Quelltext zweimal durchlaufen, bevor er den Objektcode ablegen kann. *Abb. 8.1.28* zeigt die Ausgabe des Assemblers. Alle Adressen sind mit dem Zeichen„'" gekennzeichnet. Durch addieren einer Konstanten auf diesen Adreßteil kann das Programm verschoben werden und ist dann auf einer anderen Start-Adresse lauffähig. Die Adreßteile werden von diesem Assembler in der Form MSB,LSB ausgegeben und nicht in der Form, wie sie später im Speicher stehen, nämlich LSB,MSB. Am Ende der Übersetzung wird die sogenannte Symboltabelle ausgegeben, dies ist ein Verzeichnis der verwendeten Marken (engl. Labels) und deren zugehöriger Wert.

Das Assemblerprogramm beginnt mit dem Befehl „z80". Er bewirkt, daß der Assembler die Z80-Befehle versteht, denn er kann auch 8080-Befehle verstehen, die etwas anders aussehen, obwohl sie eine Untermenge der Z80-Befehle sind. Das Programm endet mit dem Befehl „end", der dem Assembler sagt, daß hier das Programm beendet ist. Diese Befehle, die keine Befehle des Z80 sind, nennt man Pseudobefehle. Sie versteht nur der Assembler und sie sind zu dessen Steuerung gedacht.

Der Assembler besitzt eine Vielzahl von Pseudobefehlen, um die Programmierung zu erleichtern. Ein paar davon werden wir jetzt kennenlernen. *Abb. 8.1.29* zeigt die Anwendung des ORG-Befehls. Damit kann der Adreßzähler auf einen Wert gesetzt werden. Soll der Code ab Adresse 100h beginnen, so wird ein ORG 100h vorangestellt. Der Adreßzähler wird benötigt, um Sprungmarken berechnen zu können. Der Assembler hat auch die Möglichkeit, in verschiedenen Zahlensystemen zu arbeiten. Es besteht auch die Möglichkeit, unterschiedliche Verknüpfungen bei Operanden durchzuführen, so gibt es die vier Grundrechenarten und logische Verknüpfungen. In *Abb. 8.1.30* sind ein paar Operationen aufgelistet.

```
MACRO-80 3.43   27-Jul-81       PAGE    1

                                .z80

                                ;************************************
                                ;* Beispiel einer Handuebersetzung *
                                ;************************************


0000'   31 13FF                 start:  ld sp,13ffh         ;stack definieren
0003'   CD 000C'                        call init           ;forwaerts referenz
0006'   CD 0011'                        call main           ;forwaerts referenz
0009'   C3 0000'                        jp start            ;rueckwaerts referenz

000C'   3E 05                   init:   ld a,5
000E'   D3 30                           out (30h),a         ;port init
0010'   C9                              ret

0011'   06 05                   main:   ld b,5              ;schleifenzaehler
0013'   3E 01                           ld a,1              ;ausgabewert
0015'   D3 31                   loop:   out (31h),a         ;dahin ausgeben
0017'   10 FC                           djnz loop           ;rueckwaertsref
0019'   DB 31                           in a,(31h)
001B'   B7                              or a                ;test ob null
001C'   28 07                           jr z,skip           ;forwaerts referenz
001E'   3E 06                           ld a,6
0020'   D3 31                           out (31h),a
0022'   C3 0029'                        jp over             ;forwaerts referenz
0025'   3E 07                   skip:   ld a,7
0027'   D3 31                           out (31h),a
0029'   3E 08                   over:   ld a,8
002B'   D3 31                           out (31h),a
002D'   C9                              ret

                                        end


MACRO-80 3.43   27-Jul-81       PAGE    S

Macros:

Symbols:
000C'   INIT            0015'   LOOP            0011'   MAIN
0029'   OVER            0025'   SKIP            0000'   START


No Fatal error(s)
```

Abb. 8.1.28 Assemblerausgabe

```
                                  ;************************************
                                  ;* Steueranweisungen bei Assembler  *
                                  ;* ORG ANWEISUNG                     *
                                  ;************************************

                                         org 100h         ;start adresse
0100    00                               nop              ;wird auf 100h
0101    78                               ld a,b           ;abgelegt usw.
                                         org 200h         ;neuer anfang
0200    4F                               ld c,a
0201    3C                               inc a
```

Abb. 8.1.29 ORG-Anweisung

```
                                  ;************************************
                                  ;* Operanden bei Assemblern         *
                                  ;************************************

0000    3E 41                            ld a,'A'
0002    3E 50                            ld a,78+2
0004    3E 3D                            ld a,'A'-4
0006    3E 93                            ld a,10010011b
0008    3E 17                            ld a,23d
000A    3E 17                            ld a,23
000C    3E 3C                            ld a,74o
000E    3E 74                            ld a,74h
0010    3E BF                            ld a,10110101b or 00001111b
0012    3E B0                            ld a,10111010b and 11110000b
0014    3E 67                            ld a,10101011b xor 11001100b
0016    3E 09                            ld a,'9' and 0fh
```

Abb. 8.1.30 Operandenangabe beim Assembler

Marken als symbolische Operanden haben wir bei Sprungzielen schon kennengelernt. Symbolische Operanden können aber auch direkt definiert werden. *Abb. 8.1.31* zeigt wie. Durch die Verwendung von Namen wird ein Programm übersichtlicher, und die Bedeutung von Werten kann veranschaulicht werden.

Neben Befehlen müssen aber auch Daten im Speicher abgelegt werden können. Dazu gibt es eine Reihe von Pseudobefehlen, die das leisten. In *Abb. 8.1.32* sind sie dargestellt. Dabei werden drei Gruppen unterschieden. Befehle zur Ablage von Bytes (DEFB), zur Ablage von 16-Bit-Werten (DEFW) und zur Ablage von Texten (DEFM). Mit DEFB können aber auch Texte abgelegt werden. Ferner gibt es noch den nicht dargestellten Befehl DEFS. Mit ihm kann ein Speicherblock reserviert werden. Der Befehl DEFS 200 hält 200 Speicherzellen frei. Dabei wird einfach der Adreßzähler des Assemblers um 200 erhöht.

Eine Besonderheit, die nicht alle Assembler besitzen, ist die Behandlung von MAKROS. Wird eine Sequenz von Befehlen sehr oft benötigt, so verwendet man einen Unterprogrammaufruf. Doch es gibt eine ähnliche Aufgabenstellung beim Schreiben eines Programms, wo ein Unterprogrammaufruf nicht in Frage kommt. *Abb. 8.1.33* zeigt einen solchen Fall. Alle Register (nur der 8080-Satz AF,BC,DE und HL) sollen öfters auf den Stack gespeichert und dann wieder

```
                                        ;************************************
                                        ;* Steueranweisungen bei Assembler  *
                                        ;* EQU ANWEISUNG                    *
                                        ;************************************

0020                                    blank   equ     ' '     ;definitionen
000A                                    zehn    equ     10      ;zur klaren
00FE                                    port    equ     0feh    ;programmgestaltung
1234                                    konst   equ     1234h

0000    3E 20                                   ld a,blank      ;damit auch leicht
0002    D6 0A                                   sub zehn        ;aenderbar
0004    D3 FE                                   out (port),a
0006    21 1234                                 ld hl,konst
```

Abb. 8.1.31 EQU- Anweisung

```
                                        ;************************************
                                        ;* Steueranweisungen bei Assembler  *
                                        ;* SPEICHERBELEGUNGSANWEISUNGEN     *
                                        ;************************************

0000                                    tabelle:
0000    01 02 03 04                             defb 1,2,3,4
0004    61 62 63 64                             defb 'a','b','c','d'

0008                                    adressen:
0008    1000                                    defw 1000h
000A    2010                                    defw 2010h

000C                                    texte:
000C    61 6C 70 68                             defm 'alpha TEXT'
0010    61 20 54 45
0014    58 54
0016    77 69 72 64                             defm 'wird abgelegt'
001A    20 61 62 67
001E    65 6C 65 67
0022    74

                                        ; bei adressen z.B. Anwendung interrupt
                                        ; adresstabelle

                                                org 1000h       ;dort sei start
1000    0524                                    defw int1prg
1002    13FF                                    defw int2prg
1004    1400                                    defw int3prg

0524                                    int1prg equ 524h        ;z.B. dort int1
13FF                                    int2prg equ 13ffh       ;int2
1400                                    int3prg equ 1400h       ;int3
```

Abb. 8.1.32 Speicherbelegungs-Anweisung

```
                                        ;************************************
                                        ;* Steueranweisungen bei Assembler  *
                                        ;* MAKROBEFEHLE 1                   *
                                        ;************************************

                                        save    macro
                                                push af
                                                push bc
                                                push de
                                                push hl
                                                endm

                                        restore macro
                                                pop hl
                                                pop de
                                                pop bc
                                                pop af
                                                endm

                                                org 1200h

1200                                    uprg:   save
1200    F5                  +                   push af
1201    C5                  +                   push bc
1202    D5                  +                   push de
1203    E5                  +                   push hl
1204    21 1000                                 ld hl,1000h
1207    7E                                      ld a,(hl)
1208    D6 03                                   sub 3
120A    77                                      ld (hl),a
                                                restore
120B    E1                  +                   pop hl
120C    D1                  +                   pop de
120D    C1                  +                   pop bc
120E    F1                  +                   pop af
120F    C9                                      ret
```

Abb. 8.1.33 Makro-Anweisungen

zurückgeholt werden. Um sich die Schreibarbeit zu sparen und auch vor Fehlern sicherer zu sein, kann ein Makro definiert werden. Danach gibt es in unserem Beispiel zwei neue Befehle. Den Befehl SAVE und den Befehl RESTORE. Sie können, wie die anderen Mnemonics, von da an im Assemblerprogramm verwendet werden. Bei der Übersetzung wird aber der Originalcode, der in der Makrodefinition steht, übernommen und in das fertige Programmlisting einkopiert. Alle mit + gekennzeichneten Stellen sind auf diese Weise entstanden. Sie wurden vom Assembler erzeugt.

Ein anderes Beispiel für die Makrobearbeitung ist in *Abb. 8.1.34* gezeigt. Es wird dort ein Makro mit dem Namen SUBHL erzeugt. Dieser Makro besitzt aber einen Parameter, der den Namen „WERT“ trägt. Bei der Verwendung des Makros kann an Stelle des Parameters eine beliebige Konstante geschrieben werden. An allen Stellen, an denen in der Makrodefinition der Parameter „WERT“ verwendet wird, wird dann die Konstante eingesetzt.

```
                          ;*************************************
                          ;* Steueranweisungen bei Assembler  *
                          ;* MAKROBEFEHLE 2                   *
                          ;*************************************

                          subhl     macro wert
                                    push de
                                    ld de,wert
                                    xor a
                                    sbc hl,de
                                    pop de
                                    endm


                                    org 1200h
1200    2A 1000                     ld hl,(1000h)     ;laden mit inhalt 1000h
                                    subhl 395         ;subtrahieren
1203    D5               +          push de
1204    11 018B          +          ld de,395
1207    AF               +          xor a
1208    ED 52            +          sbc hl,de
120A    D1               +          pop de
120B    22 1000                     ld (1000h),hl     ;ablegen
                                    subhl 412         ;nochmals subtr
120E    D5               +          push de
120F    11 019C          +          ld de,412
1212    AF               +          xor a
1213    ED 52            +          sbc hl,de
1215    D1               +          pop de
1216    22 1002                     ld (1002h),hl     ;ablegen
```

Abb. 8.1.34 Weiteres Beispiel Makro-Anweisungen

Weitere Pseudobefehle im Assembler gestatten die bedingte Übersetzung, bei denen in Abhängigkeit von Parametern z. B. ein Programmteil übersetzt oder nicht übersetzt wird, doch würde die ausführliche Behandlung an dieser Stelle den Inhalt des Buches sprengen.

Am Schluß dieses Abschnitts sei noch der Begriff Editor kurz erwähnt, da er auch immer im Zusammenhang mit Asemblern genannt wird. Mit dem Editor ist es möglich, die Quellprogramme in den Computer einzugeben und zu verbessern. Danach steht der Quelltext der Assemblierung zur Verfügung. Liegt ein Fehler im Programm vor, z. B. ein falsch geschriebener Befehl oder eine nirgendwo vorkommende Marke, so zeigt dies der Assembler durch eine Fehlermeldung an. Der Quelltext muß dann neu editiert werden, um dann anschließend erneut übersetzt zu werden.

8.1.2 Strukturierte Programmierung

Der Begriff der strukturierten Programmierung ist heute bei den Programmierern ein Schlagwort. So wollen auch wir ein bißchen darüber hören, um die Vorteile schätzen zu lernen. Die strukturierte Programmierung ist eine Vorgehensweise, die zu leicht änderbaren und klaren Programmen führt.

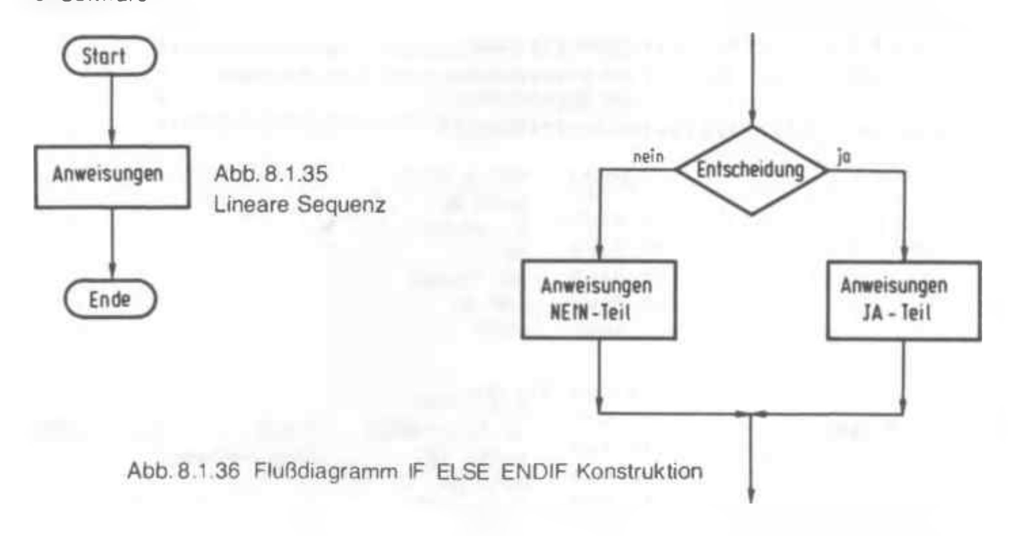


Abb. 8.1.35
Lineare Sequenz

Abb. 8.1.36 Flußdiagramm IF ELSE ENDIF Konstruktion

In der Anfangszeit der Programmierung wurde großer Wert auf besonders trickreiche Programmierung gelegt und die Programmierer haben viel Zeit darauf verwandt, Programme kompakt und optimiert zu gestalten. Genausoviel Zeit oder noch viel mehr haben sie jedoch bei der Fehlersuche gebraucht. Heute ist man davon ganz abgekommen. Die Programme sollen möglichst gut lesbar und klar verständlich im Aufbau sein. Es wurde ein Satz von Regeln entworfen, nach denen man beim Programmieren vorgehen muß, um eine klare Programmstruktur zu erhalten. Diese Regeln wurden ursprünglich für höhere Programmiersprachen entwickelt, jedoch lassen sie sich genauso auf Assemblerprogramme anwenden. Diese Regeln werden im folgenden behandelt.

Die einfachste Programmstruktur ist ein lineares Programm. Dort folgt eine Anweisung auf die nächste, ohne daß irgendwelche Sprünge ausgeführt werden. *Abb. 8.1.35* zeigt ein Flußdiagramm dazu. Bei einem Flußdiagramm gibt es ein Start und ein Endesymbol. Dies markiert einen logischen Abschnitt. In dem rechteckigen Kasten werden die Anweisungen hingeschrieben.

Mit linearer Struktur allein können aber keine sinnvollen Programme geschrieben werden. Eine Entscheidungsmöglichkeit wird gebraucht. *Abb. 8.1.36* zeigt die erste Form einer strukturierten Entscheidung. Nach Abfrage einer Bedingung wird entweder der Ja-Teil oder der Nein-Teil ausgeführt. Danach wird mit einem gemeinsamen Programmabschnitt fortgefahren. In dem Flußdiagramm ist die Entscheidung als Raute abgebildet. Als Beispiel eines Assemblerprogramms ist der Listing in *Abb. 8.1.37* gezeigt. Nach der Entscheidung folgt ein Ja-Teil und ein Nein-Teil. Nach dem Ja-Teil erfolgt im Assemblerprogramm ein unbedingter Sprung hinter den Nein-Teil. Die Bedingung ist in dem Fall, daß das Zero-Flag nicht gesetzt ist.

Eine weitere Form einer Bedingungstruktur zeigt *Abb. 8.1.38*, hier fehlt einfach der Nein-Teil. Im Fall Ja wird eine Anweisungssequenz ausgeführt, sonst nichts. *Abb. 8.1.39* zeigt die Realisierung im Assembler.

Nun gibt es aber auch andere Sprungstrukturen. *Abb. 8.1.40* zeigt die sogenannte WHILE-Bedingung. Es wird zuerst eine Abfrage durchgeführt. Ist sie nicht erfüllt, wird abgebrochen, ist sie erfüllt, so wird eine Sequenz von Anweisungen durchgeführt und dann erneut die Anweisung ausgeführt. Diese Form wird auch als Schleife bezeichnet. *Abb. 8.1.41* zeigt, wie sie im Assembler verwirklicht werden kann.

In *Abb. 8.1.42* ist eine ähnliche Form, die als REPEAT UNTIL bekannt ist, abgebildet. Es wird zunächst eine Anweisungssequenz ausgeführt und dann die Bedingung abgeprüft. Ist die Bedingung erfüllt, so wird abgebrochen, ist sie nicht erfüllt, so wird erneut die Anweisungs-

```
                        ; strukturierte Programmierung
                        ; Befehlsanordnungen

                        ; IF  ....  ELSE ....  ENDIF

                        ; if nonzero
0000'   28 05           jr z,elsepart       ;ueberspringen wenn bedingung
0002'   00               nop                ;nicht erfuellt
0003'   00               nop                ;--- beliebige instruktionen
0004'   00               nop
                        ; else
0005'   18 03           jr endif            ;uebersprung else teil
0007'                   elsepart:           ;einsprung
0007'   00               nop
0008'   00               nop                ;--- beliebige instruktionen
0009'   00               nop
                        ; endif
000A'                   endif:
000A'   00              nop
000B'   00              nop                 ;--- folge befehle
000C'   00              nop
```

Abb. 8.1.37 Assemblercodierung IF ELSE ENDIF

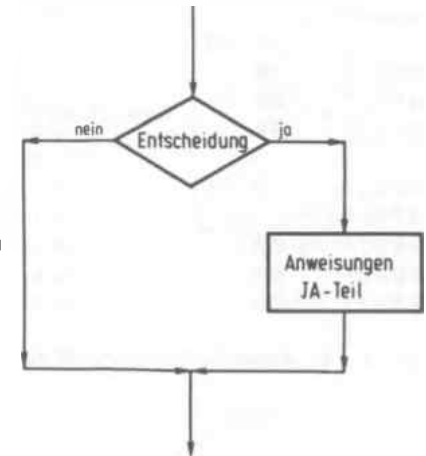


Abb. 8.1.38 Flußdiagramm IF ENDIF Konstruktion

```
                        ; IF ... ENDIF

                        ;if non zero
0000'   28 03           jr z,skip           ;ueberspringen falls zero
0002'   00               nop
0003'   00               nop                ;--- befehlssequenz
0004'   00               nop
                        ;endif
0005'                   skip:               ;uebersprungmarke
0005'   00              nop
0006'   00              nop                 ;--- folge befehle
0007'   00              nop
```

Abb. 8.1.39 Assemblercodierung IF ENDIF

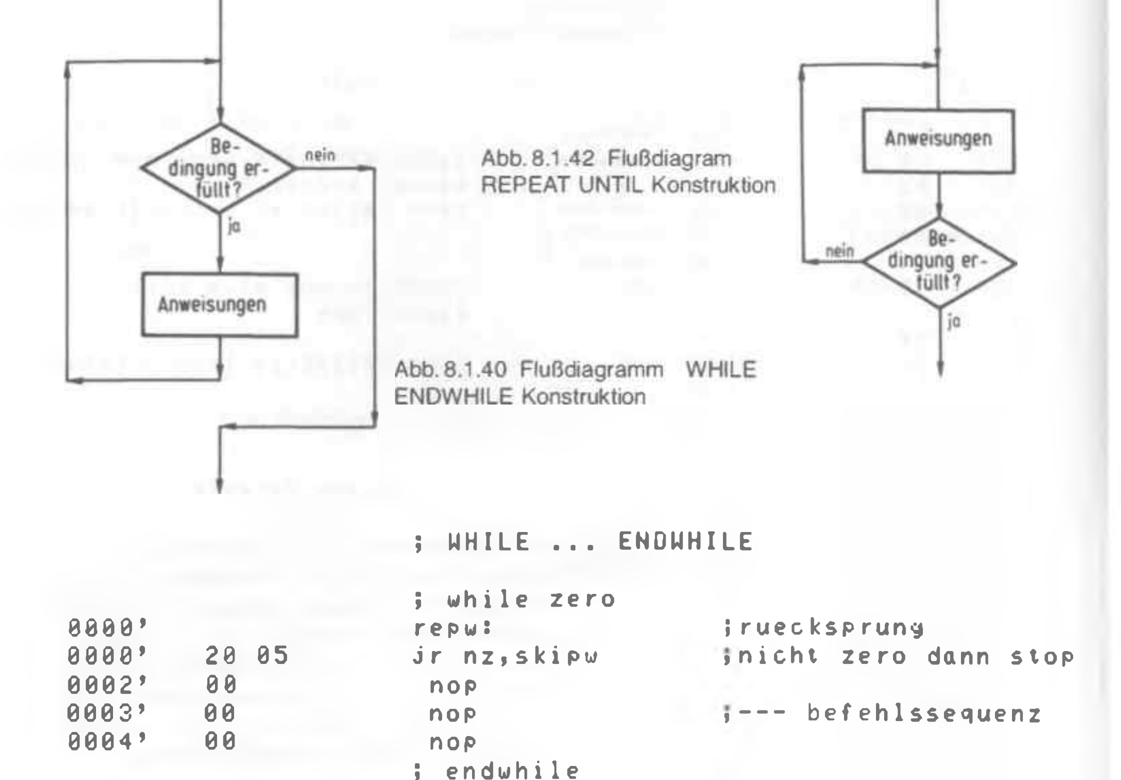


Abb. 8.1.42 Flußdiagram REPEAT UNTIL Konstruktion

Abb. 8.1.40 Flußdiagramm WHILE ENDWHILE Konstruktion

```
                       ; WHILE ... ENDWHILE

                       ; while zero
0000'                  repw:              ;ruecksprung
0000'    20 05         jr nz,skipw        ;nicht zero dann stop
0002'    00             nop
0003'    00             nop               ;--- befehlssequenz
0004'    00             nop
                       ; endwhile
0005'    18 F9         jr repw            ;immer zurueck hier
0007'                  skipw:             ;ende teil
0007'    00            nop
0008'    00            nop                ;--- folge befehle
0009'    00            nop
```

Abb. 8.1.41 Assemblercodierung WHILE ENDWHILE

sequenz durchgeführt. *Abb. 8.1.43* zeigt das Assemblerprogramm. Der Unterschied zwischen WHILE und REPEAT besteht darin, daß bei REPEAT die Anweisungssequenz mindestens einmal durchlaufen wird und bei WHILE der Abbruch erfolgen kann, ohne daß die Anweisungssequenz auch nur einmal durchlaufen wurde.

Eine oft benötigte Schleifenstruktur ist die, bei der ein Zähler, der sogenannte Schleifenzähler mit auftritt. *Abb. 8.1.44* zeigt das Flußdiagramm. Dabei ist die Struktur im Prinzip identisch mit der Form REPEAT UNTIL. *Abb. 8.1.45* zeigt das Assemblerprogramm mit einem 8-Bit-Zähler, *Abb. 8.1.46* zeigt die Realisierung, wenn das Register B verwendet wird und in *Abb. 8.1.47* ist die Lösung mit einem 16-Bit-Zähler beschrieben. Häufig wird auch die Konstruktion einer Endlosschleife mit mehreren Abbruchkritierien gebraucht. *Abb. 8.1.48* zeigt die Lösung. Die Abfragen sind diesmal inmitten der Programmabschnitte. In *Abb. 8.1.49* ist das dazugehörige Assemblerprogramm beschrieben.

```
                        ; REPEAT ... UNTIL

                        ; repeat
0000'                   rep:                ;einsprung fuer rueckkehr
0000'    00              nop
0001'    00              nop                ;--- befehlssequenz
0002'    00              nop
                        ; until carry
0003'    30 FB          jr nc,rep           ;nicht erfuellt dann zurueck
0005'    00             nop
0006'    00             nop                 ;--- folge befehle
0007'    00             nop
```

Abb. 8.1.43 Assemblercodierung REPEAT UNTIL

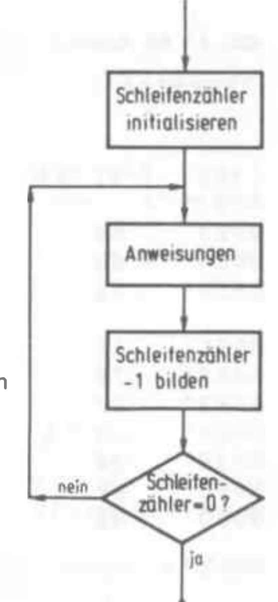


Abb. 8.1.44 Flußdiagramm der Schleifenkonstruktion

```
                          org 0


                        ; DO register,anzahl ... ENDDO

                        ; DO c,5
0000'    0E 05          ld c,5
0002'                   lpp:                ;schleifenruecksprung
0002'    00              nop
0003'    00              nop                ;--- befehlssequenz
0004'    00              nop
                        ; ENDDO
0005'    0D             dec c
0006'    20 FA          jr nz,lpp           ;bis c=0 wiederholen
0008'    00             nop
0009'    00             nop                 ;--- folge befehle
000A'    00             nop
```

Abb. 8.1.45 Assemblercodierung der DO-Schleife mit dem C-Register

```
                                    ; DO b,6
0000'   06 06                       ld b,6
0002'                               dolpl:              ;schleifenruecksprung
0002'   00                           nop
0003'   00                           nop
0004'   00                           nop
                                    ; ENDDO
0005'   10 FB                       djnz dolpl          ;sonderfall direktbefeh
0007'   00                          nop
0008'   00                          nop
0009'   00                          nop

                                            end
```

Abb. 8.1.46 Assemblercodierung der DO-Schleife mit dem B-Register

```
                                    ; DO registerpaar,startwert ... ENDDO

                                    ; do de,1000
0000'   11 03E8                     ld de,1000
0003'                               dolp:               ;ruecksprungmarke
0003'   00                           nop
0004'   00                           nop                ;--- befehlssequenz
0005'   00                           nop
                                    ; enddo
0006'   1B                          dec de
0007'   7B                          ld a,e
0008'   B2                          or d                ;test ob 0
0009'   20 F8                       jr nz,dolp          ;nein dann nochmals
000B'   00                          nop
000C'   00                          nop                 ;--- befehls folge
000D'   00                          nop
```

Abb. 8.1.47 Assemblercodierung der DO-Schleife mit dem DE-Register

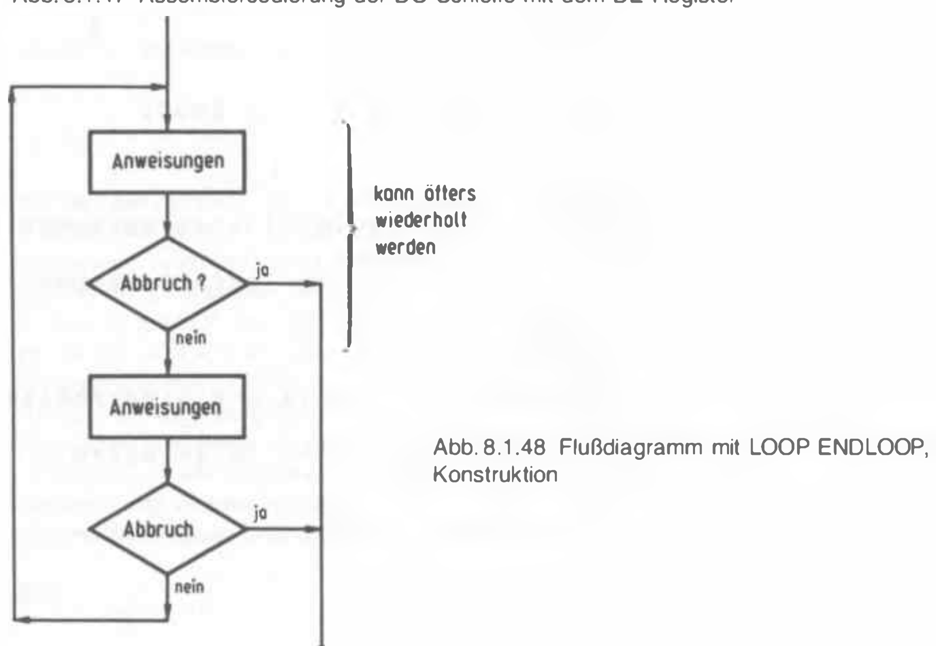


Abb. 8.1.48 Flußdiagramm mit LOOP ENDLOOP, Konstruktion

```
                        ; LOOP , EXIT ,  ENDLOOP

                        ; loop
0000'                   loop:                 ;schleifenruecksprung
0000'   00               nop
0001'   00               nop                  ;--- befehlssequenz
0002'   00               nop
                         ;exitif zero
0003'   28 0A            jr z,fina
0005'   00               nop
0006'   00               nop                  ;--- befehlssequenz
0007'   00               nop
                         ;exitif no carry
0008'   30 05            jr nc,fina
000A'   00               nop
000B'   00               nop                  ;--- befehlssequenz
000C'   00               nop
                        ; endloop
000D'   18 F1           jr loop               ;ruecksprung
000F'                   fina:                 ;ende der schleife
000F'   00              nop
0010'   00              nop                   ;--- folge befehle
0011'   00              nop

                        end
```

Abb. 8.1.49 Assemblercodierung mit LOOP ENDLOOP

Diese einzelnen Module können zu einem gesamten Programm geformt werden. Dabei kann anstelle des Begriffs Anweisung wieder irgendeine der Formen stehen. Sie können damit beliebig verschachtelt werden. Beispiel:

```
loop
Anweisung
if zero
  Anweisung
 else
  Anweisung
 repeat
   Anweisung
 until no carry
  Anweisung
 endif
 exitif carry
 Anweisung
endloop
Anweisung
```

IF..ELSE..ENDIF wurde als Abkürzung für die Entscheidungskonstruktion verwendet, LOOP ENDLOOP zusammen mit EXITIF für die Endlosschleife und REPEAUT UNTIL für die

```
                        ; geschachtelte konstruktion

                        ;loop
0000'                   lpp:
0000'   00               nop
0001'   00               nop
0002'   00               nop
                         ;if zero
0003'   20 05            jr nz,sk1
0005'   00                nop
0006'   00                nop
0007'   00                nop
                         ;else
0008'   18 0B            jr sk2
000A'                    sk1:
000A'   00                nop
000B'   00                nop
000C'   00                nop
                          ;repeat
000D'                     rp1:
000D'   00                 nop
000E'   00                 nop
000F'   00                 nop
                          ;until no carry
0010'   38 FB             jr c,rp1
0012'   00                nop
0013'   00                nop
0014'   00                nop
                         ;endif
0015'                    sk2:
                         ;exitif carry
0015'   38 05            jr c,fina
0017'   00               nop
0018'   00               nop
0019'   00               nop
                        ;endloop
001A'   18 E4           jr lpp
001C'                   fina:
001C'   00              nop
001D'   00              nop
001E'   00              nop
```

Abb. 8.1.50 Geschachtelte Konstruktion

Wiederholungsschleife. Die Anweisungen innerhalb der Konstruktionen werden jeweils um einen Platz eingerückt, was besonders gut die Struktur des Programms darstellt. An Stelle des Begriffs Anweisung können beliebige Z80-Befehle stehen. Dabei dürfen sie keine Sprünge mehr enthalten, wohl aber z. B. Unterprogrammaufrufe. *Abb. 8.1.50* zeigt das Assemblerprogramm dazu.

Beim Entwurf eines Programms kann man so vorgehen, daß zuerst mit Hilfe der Begriffe IF ELSE ENDIF etc. eine grobe Struktur des Programms aufgezeichnet wird. Dann kann entweder direkt codiert werden oder es werden die einzelnen Flußdiagramme anstelle der Struktur-Bezeichnungen gezeichnet. Es kann aber auch mit dem Zeichnen des Flußdiagramms begonnen

werden. Wichtig ist dann, daß nur die vorgezeigten Strukturen verwendet werden und keine anderen, da sich sonst eventuell ein nicht strukturiertes Programm ergibt. Im nächsten Abschnitt ist noch ein komplettes Beispiel abgedruckt, nämlich das Monitorprogramm, das nach diesen Regeln entworfen wurde.

8.2 Das Grundprogramm

Im Verlauf der vorherigen Kapitel wurde schon oft mit dem Grundprogramm gearbeitet. Hier soll nun eine kleine Zusammenfassung erfolgen.

Abb. 8.2.1 zeigt die verschiedenen Menüs. Nach einem Reset gelangt man in das Hauptmenü. Von dort aus kann man folgende Funktionen aufrufen:

aendern:
Eingabe eines Programms in Maschinensprache oder ändern eines bestehenden Programms.

starten:
Starten eines Programms. Dazu wird die Startadresse angegeben.

ansehen:
Ansehen eines Speicherbereiches.

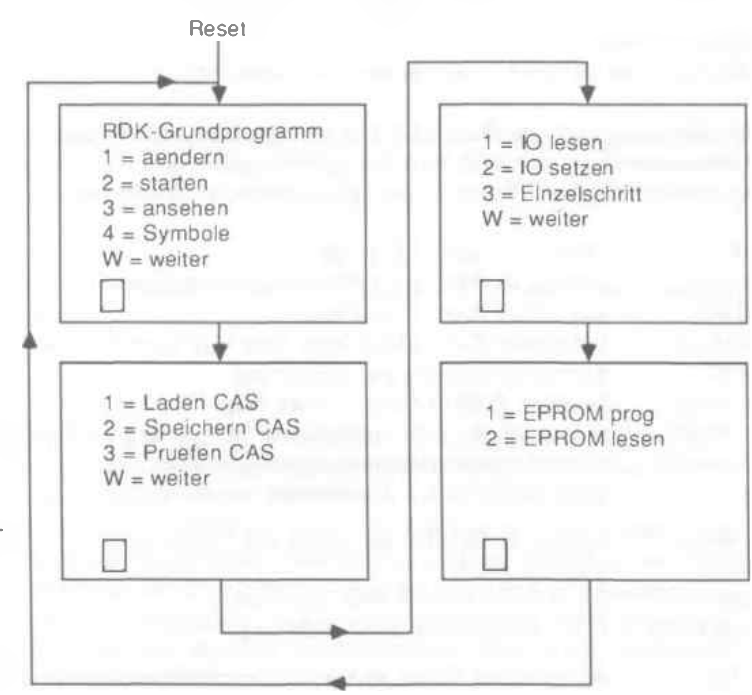


Abb. 8.2.1
Abfolge der Grundprogramm-Menüs

Symbole:
Ausgabe aller definierten Symbole. Dabei werden die neu definierten Symbole ausgegeben. Mit „W = weiter“ gelangt man zum nächsten Menü.

Laden CAS:
Laden einer Programm- oder Daten-Datei von einer Cassette.

Speichern CAS:
Speichern eines Programms oder von Daten.

Pruefen CAS:
Vergleich eines abgespeicherten Programms oder Daten mit dem aktuellen Speicherinhalt. Das nächste Menü beinhaltet Testfunktionen:

IO lesen:
Lesen von IO-Ports und Ausgabe auf dem Bildschirm für Testzwecke.

IO setzen:
Ausgabe von Werten an IO-Ports für Testzwecke.

Einzelschritt:
Schrittweise Abarbeitung von Maschinenbefehlen zum Programmtest. Im letzten Menü hat man die Möglichkeit EPROMs zu programmieren:

EPROM prog:
Programmieren eines EPROMs mit der PROMMER-Baugruppe.

EPROM lesen:
Einlesen eines EPROM-Inhalts in den Arbeitsspeicher.

Bei allen Angaben von Zahlen hat man im Grundprogramm verschiedene Möglichkeiten, die Unterscheidung ob Byte oder Wort, wird allerdings nur bei der Dateneingabe im Menü „aendern“ ausgewertet und beeinflußt dort das Fortschreiten der Speicheradresse.

21	sedezimale Zahl, 1 Byte lang
5.W	sedezimale Zahl, auf 1 Wort = 2 Byte erweitert
4FA2	sedezimale Zahl, 1 Wort lang
4FA2.B	sedezimale Zahl, 1 Byte lang, Rest wird abgeschnitten
#84	dezimale Zahlenangabe, 1 Byte lang
#84.W	dezimale Zahlenangabe, 1 Wort lang
−#56	negative Zahl, wird umgerechnet, 1 Wort lang (wegen 2er Kompl.)
AA+BB	Summe zweier sedezimaler Zahlen, 1 Wort lang
$	Stand des aktuellen Adreßzählers bei der Eingabezeile
alpha:=3A5	alpha wird definiert und erhält den Wert 3A5
alpha	alpha als Wert
alpha:=$	die aktuelle Adresse wird zugewiesen
alpha:=%	alpha wird aus der Symboltabelle gelöscht
"Test"	Ablage eines Textes in ASCII, hier vier Speicherzellen lang

Tabelle 8.2.1 Zusammenstellung der Grundprogrammbefehle

Befehl	Beschreibung
SCHREITE	HL= Anzahl der Schritte
SCHR16TEL	HL = Anzahl der Schritte in 1/16 Punkt
DREHE	HL = Winkel in Grad
HEBE	Danach keine Schreibspur mehr
SENKE	Danach wieder Schreibspur aktiv
SCHLEIFE	Schleifenanfang , HL = Zahl der Durchläufe
ENDSCHLEIFE	Schleifenende, Klammer zu SCHLEIFE
SET	HL = x-Koordinate (0..511) DE = y-Position (0..511) wird umgerechnet BC = Startwinkel in Grad
MOVETO	HL = x-Koordinate(0..511) DE = y-Koordinate(0..255), realer Wert
DRAWTO	HL = x-Koordinate(0..511) DE = y-Koordinate(0..255), realer Wert
WRITE	HL = Adresse des Textblockes
READ	HL = Adresse des Versorgungsblocks C = 1, mit Umrandung; C=0, ohne Umrandung
CI	Einlesen eines Zeichens nach A
CSTS	A = 0FFh, wenn ein Zeichen da
RI	Einlesen eines Zeichens von CAS nach A
PO	Ausgabe eines Zeichens von A nach CAS
CLR	Löschen aller vier Bildseiten
CLPG	Löschen der aktuellen Schreibseite
WAIT	Warten, bis GDP fertig
RAM	Adresse des verwendbaren RAMs, normalerweise 8800h

Bei der Eingabe von Daten im Menü „aendern" können auch mehrere Werte nebeneinanderstehen, z. B.:

21 34.W

Wenn man ein Minuszeichen beim zweiten Operanden verwenden will, so muß man allerdings davor ein Komma setzen, also:

21,−34.W

sonst wird die Differenz zwischen 21h und 34h gebildet und dieses Ergebnis abgespeichert.

Das Grundprogramm besitzt eine Reihe vorgefertigter Unterprogramme, die in *Tabelle 8.2.1* zusammengefaßt sind.

Der Bildschirm der GDP besitzt eine Auflösung von 512 × 256 Bildpunkten. Das ergibt ein Seitenverhältnis 2:1. Alle Befehle der Schildkrötengrafik rechnen mit einer Auflösung von 512 × 512 und nehmen die Umrechnung auf den Bildschirmn automatisch vor. Alle Koordinaten werden zusätzlich mit 16facher Genauigkeit gespeichert, um Kreisberechnungen etc. exakt zu ermöglichen. Die GDP erlaubt einen Adreßraum von 4096 × 4096 Punkten. Alle Punkte, die nicht im Fenster von 512 × 256 liegen, bleiben daher unsichtbar.

Die Befehle:

SCHREITE:
Im Registerpaar HL steht die Anzahl der Schritte in Bildpunkten (gilt nur bei horizontaler Blickrichtung). Negative Zahlenangaben lassen die Schildkröte rückwärts schreiten. Die Schildkröte schreitet außerdem in die aktuelle Blickrichtung. Es wird mit einer Auflösung von 512 × 512 Bildpunkten gerechnet. Die neue Position wird in 16facher Genauigkeit gespeichert.

SCHR16TEL:
Wie oben, jedoch mit 1/16 Schritt. Dieser Befehl wird benötigt, um z. B. Kreise mit beliebigem Durchmesser zu zeichnen.

DREHE:
Im Registerpar HL steht der Winkel in Grad. Positive Zahlenangaben drehen die Schildkröte gegen den Uhrzeigersinn.

HEBE:
Danach ist keine Schreibspur mehr sichtbar, die Schildkröte bewegt sich aber noch weiter.

SENKE:
Bewegungen der Schildkröte hinterlassen wieder eine Schreibspur.

SCHLEIFE:
Schleifenanfang. Im Registerpaar HL steht die Anzahl der Durchläufe. Es werden Rücksprungadresse und Schleifenzähler auf dem Stack aufbewahrt.

ENDSCHLEIFE:
Ende einer Schleife. Jede Schleife muß durch SCHLEIFE . . . ENDSCHLEIFE geklammert sein.

SET:
HL = x-Position (0 . . 511), DE = y-Position (0 . . 511), BC = Startwinkel der Schildkröte in Grad. Damit wird eine neue Schildkrötenposition absolut festgesetzt. Die Schildkröte zeichnet auch eine Linie zu diesem neuen Punkt, wenn man das mit HEBE nicht zuvor verhindert hat.

MOVETO:
Absolutes Positionieren für die GDP. HL = ×-Koordinate (0 . . 511), DE = y-Koordinate (0 . . 255).

DRAWTO:
Eine Linie wird vom letzten MOVETO-Punkt zum neuen Punkt gezeichnet. HL = x-Koordinate (0. . . 511), DE = y-Koordinate (0 . . 255).

WRITE:
Ausgabe eines Textes auf dem Bildschirm.
HL = Adresse des Textblockes.
Der Textblock wird wie folgt aufgebaut:
x.W y.W Höhe.B Schräge.B "Text" 0

Beispiel:

```
TEXT1:=$
20.W 30.W 44.B 0.B
"Hallo Test"
00
START:
21 TEXT1.W
CD WRITE
C9
```

Wenn man das Programm startet, so erscheint der Text „Hallo Test“ auf dem Bildschirm. Der Text wird beginnend bei der Position x = 20 und y = 30 mit der Schriftgröße 4 ausgegeben.

READ:
Eingabe eines Textes. HL = Adresse des Versorgungsblockes,
C = 1 mit Umrandung, C = 0 ohne Umrandung.
Versorgungsblock:
x.W y.W Höhe.B 0.B max.B aktuell.B – – reservierter Speicher – –
Beispiel:

```
21 8900.W
0E 01
CD READ
...
8900:
10.W 20.W 22.B 0 6.B 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Nach Aufruf erscheint ein Rahmen auf dem Bildschirm. Nun kann ein beliebiger Text mit bis zu 6 Zeichen (je nach max.B) eingegeben werden. Nach Eingabe von CR (carriage return) verschwindet der Rahmen. Der Text ist anschließend im reservierten Speicherbereich abgelegt und die Variable akt.B enthält die Anzahl der tatsächlich eingelesenen Zeichen.

CI:
Eingabe eines Zeichens von der Tastatur in das Register A. Das Unterprogramm wartet solange, bis ein Zeichen eingegeben wird.

CSTS:
Prüft ob ein Zeichen eingegeben wurde. Wenn ja, so wird der Wert 0FFh im Register A übergeben, sonst der Wert 0. Wenn ja, so kann man anschließend mit CI das Zeichen einlesen, ohne warten zu müssen.

RI:
Einlesen eines Zeichens von der CAS-Baugruppe in das Register A. Es wird solange gewartet, bis ein Zeichen eingelesen wird.

PO:
Ausgabe eines Zeichens vom Register C an die CAS-Baugruppe.

CLR:
Löschen des Bildschirms. Alle vier Bildschirmseiten werden gelöscht.

CLPG:
Löschen der aktuellen Schreibseite. Diese kann auch unsichtbar sein.

WAIT:
Warten bis der Grafik-Prozessor mit der Abarbeitung eines Befehls fertig ist. Danach kann man z. B. auch Werte direkt an die GDP-Register (70h . . 7Fh) ausgeben.

Ebenfalls ist es möglich, das Seitenauswahlregister Port 60h zu ändern. Damit kann man die Schreib- und Leseseiten ändern.
Beispiel löschen der zweiten Bildschirmseite (bei 1 . . 4 Seiten).

```
CD WAIT
3E 40
D3 60
CD CLPG
```

Die erste Seite bleibt sichtbar, die zweite wird gelöscht.

Alle diese Befehle stehen auch als Einsprung zur Verfügung. Wenn man eigene Programme schreibt, die von der Version des Grundprogramms unabhängig sein sollen, so empfiehlt es sich, nur diese Einsprünge zu verwenden. Wenn man das Grundprogramm zur Programmeingabe verwendet, so werden die symbolisch angegebenen Namen automatisch in Aufrufe dieser Einsprünge übersetzt. *Abb. 8.2.2* zeigt die Einsprungtabelle, so wie man sie mit Hilfe des Disassemblers (siehe nächster Abschnitt) erhält.

Zwei Adressen haben noch eine besondere Bedeutung. Die Adresse 38h und 66h. Beim Z80 erfolgt der Interrupt des Mode IM1 zum Beispiel direkt auf die Adresse 38h. Um diesen Interrupt verwenden zu können, befindet sich im Grundprogramm an dieser Stelle ein Sprung auf die Adresse 8003h. Dort sind 3 Bytes für einen weiteren Sprung reserviert. Man kann dorthin den Sprung auf ein eigenes Interrupt-Programm legen. Die Adresse 66h ist die Einsprungadresse bei einem NMI-Interrupt, der durch die Leitung NMI ausgelöst wird. Ein Sprung führt auf die Adresse 8000h, und dorthin kann man einen Sprung zum eigenen Interrupt-Programm hinschreiben. Dafür sind ebenfalls drei Bytes resereniert. Man kann damit zum Beispiel eine Uhr realisieren.

8.2.1 Kleine Beispiele:

Vielstar:
Abb. 8.2.3 zeigt das Programm und das Ergebnis auf dem Bildschirm. Das Programm besteht aus zwei ineinander geschachtelten Schleifen.

Aufgaben: Wie sieht die Figur der inneren Schleife aus? Versuchen Sie das Programm nachzuvollziehen.

Hinweis: Im Einzelschritt mit dem Computer geht es leichter.

Vielleiter:
Abb. 8.2.4 zeigt das Programm. Hier sind drei Schleifen verschachtelt. Die innerste Schleife zeichnet ein kleines Quadrat. Die nächste Schleife setzt acht solcher Quadrate aneinander um eine Leiter zu erhalten und die äußerste Schleife zeichnet 36 Leitern um jeweils 10° gedreht.

```
0000 C3381F      JP 1F38H
0003 C36B04    SCHREITE: JP 046BH
0006 C3C004    DREHE: JP 04C0H
0009 C3E404    HEBE: JP 04E4H
000C C3EB04    SENKE: JP 04EBH
000F C3D803    SCHLEIFE: JP 03D8H
0012 C3F903    ENDSCHLEIFE: JP 03F9H
0015 C34604    SET: JP 0446H
0018 C35501    MOVETO: JP 0155H
001B C36501    DRAWTO: JP 0165H
001E C3E005    WRITE: JP 05E0H
0021 C38506    READ: JP 0685H
0024 C36900    CI: JP 0069H
0027 C34100    CSTS: JP 0041H
002A C3E500    RI: JP 00E5H
002D C3EE00    PO: JP 00EEH
0030 C31F01    CLR: JP 011FH
0033 C34F02    CLPG: JP 024FH
0036 00         NOP
0037 00         NOP
0038 C30380     JP 8003H
003B C3F800    WAIT: JP 00F8H
003E C36E04    SCHR16TEL: JP 046EH
-

0066 C30080      JP 8000H
-
```

Abb. 8.2.2 Einsprünge in das Grundprogramm, wichtige Adressen

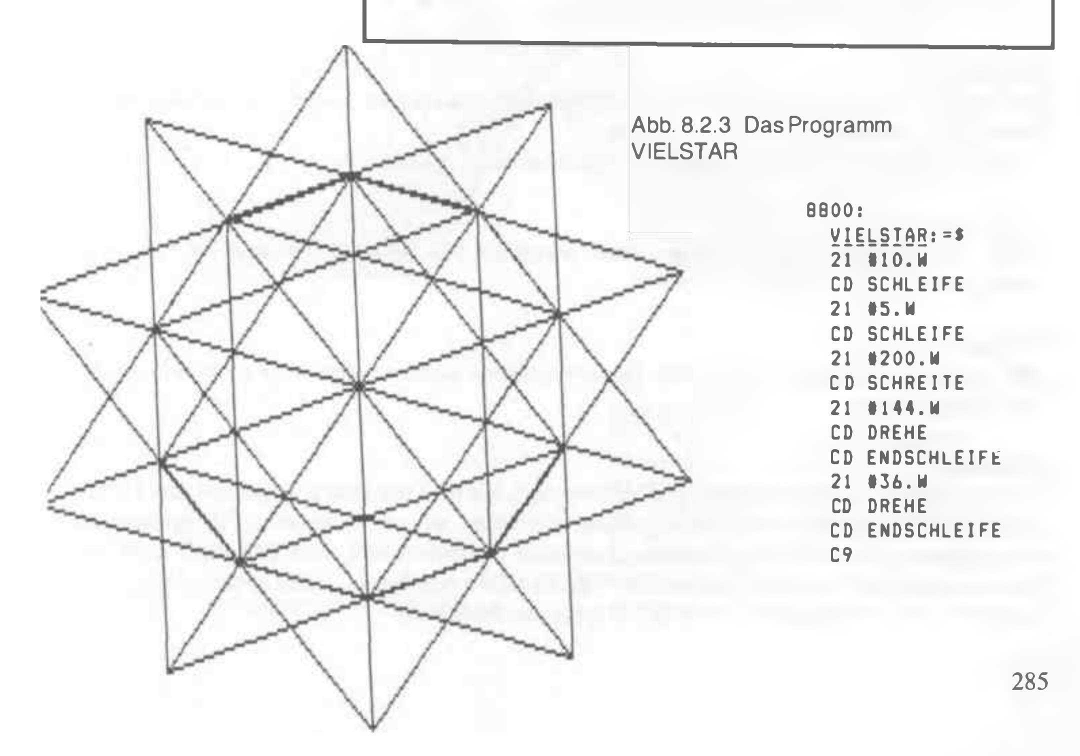
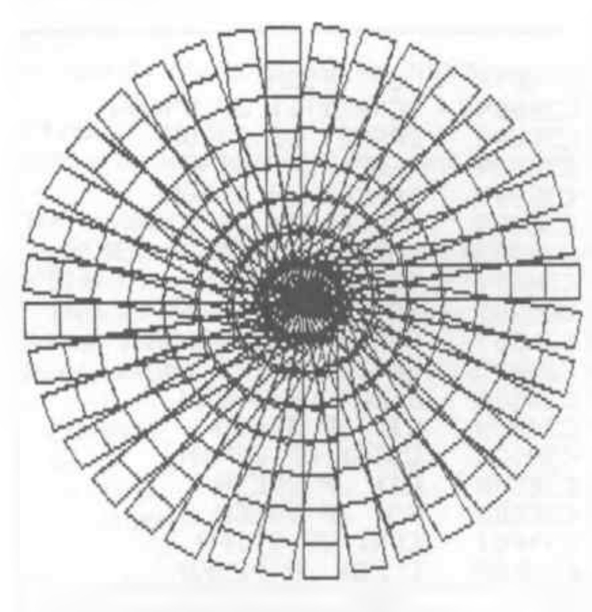


Abb. 8.2.3 Das Programm VIELSTAR

```
8800:
  VIELSTAR:=$
  21 #10.W
  CD SCHLEIFE
  21 #5.W
  CD SCHLEIFE
  21 #200.W
  CD SCHREITE
  21 #144.W
  CD DREHE
  CD ENDSCHLEIFE
  21 #36.W
  CD DREHE
  CD ENDSCHLEIFE
  C9
```



```
8800:
  VIELLEITER:=$
  21 #36.W
  CD SCHLEIFE
  21 #8.W
  CD SCHLEIFE
  21 #4.W
  CD SCHLEIFE
  21 #20.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  CD ENDSCHLEIFE
  21 #20.W
  CD SCHREITE
  CD ENDSCHLEIFE
  21 -#160.W
  CD SCHREITE
  21 #10.W
  CD DREHE
  CD ENDSCHLEIFE
  C9
```

Abb. 8.2.4 Das Programm VIELLEITER

Mehrkreise:
Wenn man zusätzliche Variable verwendet, kann man interessantere Figuren zeichnen. *Abb. 8.2.5* zeigt das Programm. Um Kreise beliebiger Größe zu erhalten, wird hier das Unterprogramm SCHR16TEL verwendet.

Die Speicherzelle 8900h beinhaltet die Variable.

Groessen:
Das Programm in *Abb. 8.2.6* ist ähnlich zum vorherigen aufgebaut. Finden Sie die Gemeinsamkeiten und Unterschiede.

Entwerfen Sie neue Figuren anhand der gefundenen Unterschiede.

Vspiraleb:
Wenn man zusätzlich zur Vergrößerung einer Grundfigur eine Drehung programmiert, erhält man Figuren wie *Abb. 8.2.7* zeigt.

Schienen:
Daß man aber nicht nur künstlerische Figuren zeichnen kann, sondern auch praktische Dinge, zeigt *Abb. 8.2.8*.

Startehier:
Das Programm mit dem merkwürdigen Namen zeigt, wie man mit Texten umgehen kann. Dunkle Schrift (auf dem Bildschirm) auf hellem Grund erhält man, wenn man ein helles Feld zeichnet und anschließend eine Schrift mit gesetztem Löschstift darüberschreibt. Den Löschstift kann man durch Ausgabe des Befehls 01 an den Port 70h des GDPs einschalten. Siehe auch das Kapitel, in dem die GDP beschrieben ist. *Abb. 8.2.9* zeigt das Programm.

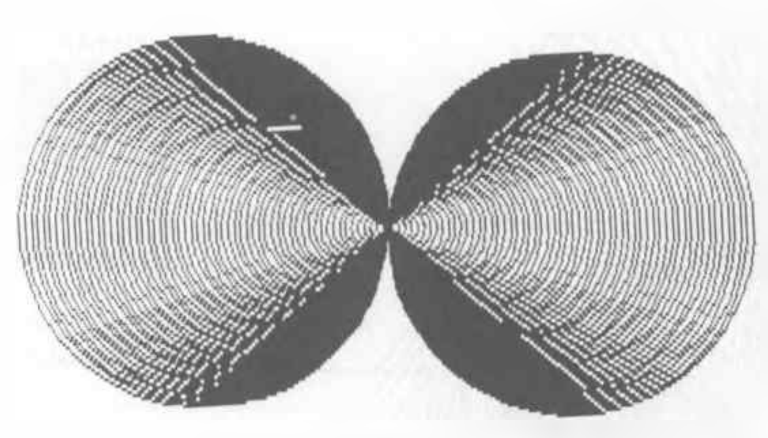


Abb. 8.2.5 Das Programm MEHRKREISE

```
8800:
 MEHRKREISE:=$
  21 #300.W
  22 8900.W
  21 #120.W
  CD SCHLEIFE
  21 #36.W
  CD SCHLEIFE
  2A 8900.W
  CD SCHRIGTEL
  21 #10.W
  CD DREHE
  CD ENDSCHLEIFE
  2A 8900.W
  11 -#5.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```

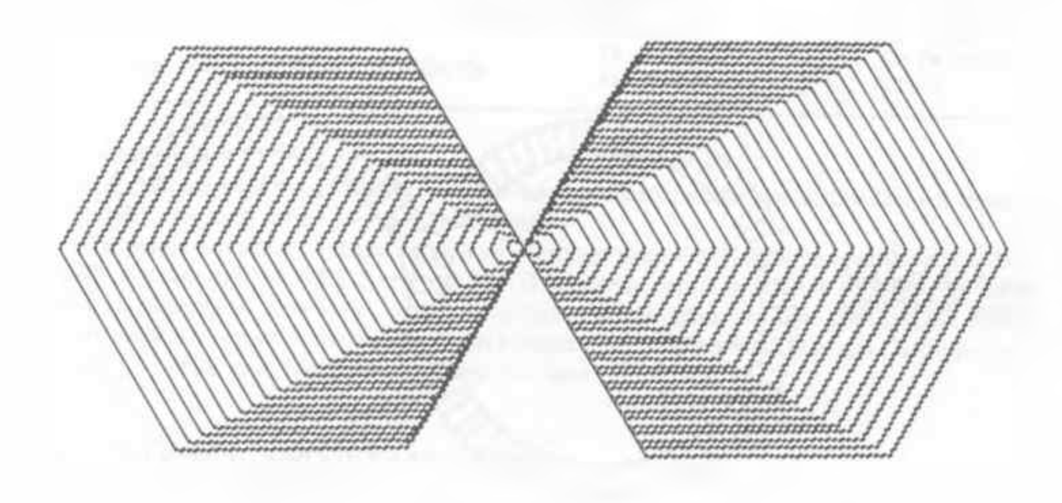


Abb. 8.2.6 Das Programm GROESSEN

```
8800:
  GROESSEN:=$
  21 #140.W
  22 8900.W
  21 #56.W
  CD SCHLEIFE
  21 #6.W
  CD SCHLEIFE
  2A 8900.W
  CD SCHREITE
  21 #60.W
  CD DREHE
  CD ENDSCHLEIFE
  2A 8900.W
  11 -#5.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```

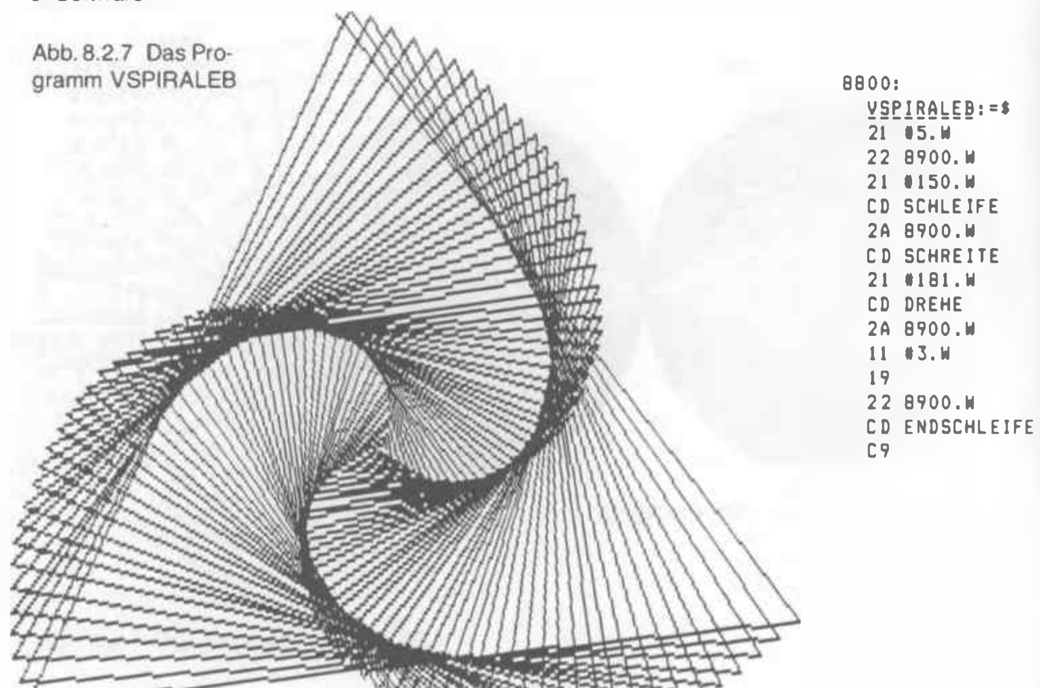


Abb. 8.2.7 Das Programm VSPIRALEB

```
8800:
  VSPIRALEB:=$
  21 #5.W
  22 8900.W
  21 #150.W
  CD SCHLEIFE
  2A 8900.W
  CD SCHREITE
  21 #181.W
  CD DREHE
  2A 8900.W
  11 #3.W
  19
  22 8900.W
  CD ENDSCHLEIFE
  C9
```

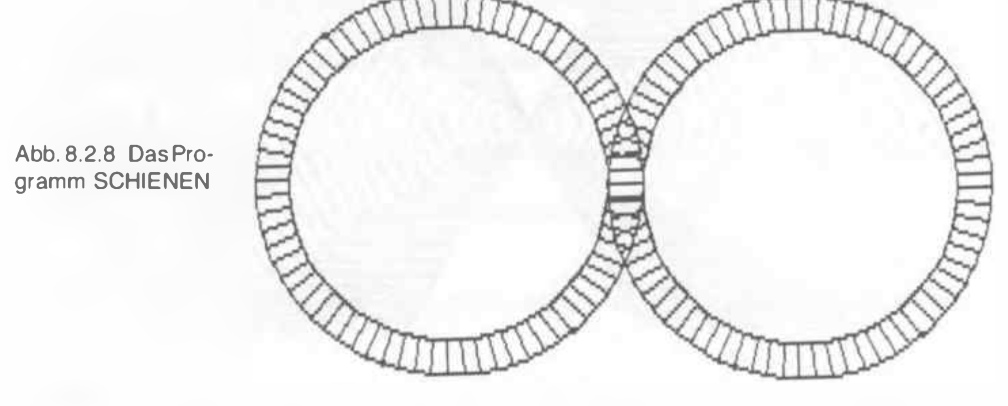
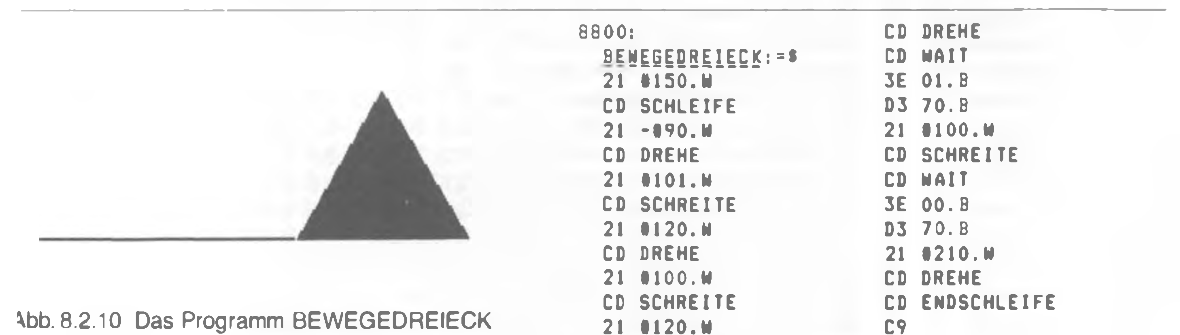


Abb. 8.2.8 Das Programm SCHIENEN

```
8800:
  SCHIENEN:=$
  21 #2.W
  CD SCHLEIFE
  21 #72.W
  CD SCHLEIFE
  21 #8.W
  CD SCHREITE
  21 -#8.W
  CD SCHREITE
  21 -#90.W
  CD DREHE
  21 #20.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  21 #11.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  CD HEBE
  21 #20.W
  CD SCHREITE
  21 -#90.W
  CD DREHE
  21 -#3.W
  CD SCHREITE
  CD SENKE
  21 #5.W
  CD DREHE
  CD ENDSCHLEIFE
  21 #90.W
  CD DREHE
  21 -#20.W
  CD SCHREITE
  21 #90.W
  CD DREHE
  CD ENDSCHLEIFE
  C9
```

```
8800:
 TEXT:=$
 #160.W
 #130.W
 33.B
 0.B
 "HALLO"
 0.B
 TEXT1:=$
 #300.W
 #130.W
 33.B
 0.B
 "HALLO"
 0.B
STARTEHIER:=$
 21 TEXT1
 CD WRITE
 21 #60.W
 CD SCHLEIFE
 21 #120.W
 CD SCHREITE
 21 #90.W
 CD DREHE
 21 #1.W
 CD SCHREITE
 21 #90.W
 CD DREHE
 21 #120.W
 CD SCHREITE
 21 -#90.W
 CD DREHE
 21 #1.W
 CD SCHREITE
 21 -#90.W
 CD DREHE
 CD ENDSCHLEIFE
 CD WAIT
 3E 01.B
 D3 70.B
 21 TEXT
 CD WRITE
 C9
```

Abb. 8.2.9 Das Programm STARTEHIER



```
8800:
BEWEGEDREIECK:=$
 21 #150.W
 CD SCHLEIFE
 21 -#90.W
 CD DREHE
 21 #101.W
 CD SCHREITE
 21 #120.W
 CD DREHE
 21 #100.W
 CD SCHREITE
 21 #120.W
 CD DREHE
 CD WAIT
 3E 01.B
 D3 70.B
 21 #100.W
 CD SCHREITE
 CD WAIT
 3E 00.B
 D3 70.B
 21 #210.W
 CD DREHE
 CD ENDSCHLEIFE
 C9
```

Abb. 8.2.10 Das Programm BEWEGEDREIECK

Bewegedreieck:
Abb. 8.2.10 zeigt ein Beispiel dafür, wie man bewegte Grafik machen kann. Das Dreieck bewegt sich von links nach rechts über den Bildschirm.

Für anspruchsvollere Grafiken sollte man allerdings nicht die Schildkrötengrafik verwenden, sondern direkt mit MOVETO und DRAWTO arbeiten. Ferner empfiehlt es sich z. B. zwei Bildschirmseiten zu verwenden. In der einen baut man unsichtbar das neue Bild auf, die andere wird in der Zwischenzeit dargestellt. Dann schaltet man um (Port 60h), und das Ganze beginnt erneut. So erhält man flickerfreie bewegte Grafiken.

8.2.2 Das Grundprogrammlisting

Leider würde es den Umfang des Buches sprengen, das Assemblerlisting des Grundprogramms vollständig abzudrucken. Wer daran interessiert ist, kann es vom Verlag oder der Bausatzfirma (siehe Anhang) beziehen.

Hier ist nur der Hexdump abgedruckt, um eine einfache Hilfe darzustellen. Das Grundprogramm ist auch in EPROM-Form beim Bausatzhersteller zu bekommen, wer aber will, kann den Hexdump in ein EPROM-Progammiergerät eintippen. Achtung: um Fehler zu vermeiden, wurde eine Prüfsumme für jede Zeile berechnet, die immer über alle Datenbytes gebildet wurde (zusammenzählen der Werte, ohne einen 16-Bit-Überlauf zu berücksichtigen). Man sollte die Prüfsummen unbedingt mit selbst gebildeten vergleichen.

Abb. 8.2.11 zeigt den Hexdump des Grundprogramms der Version 2.0.

```
-------------- Seite  1 ------- Datei Grund 2.0 -----------
Rom                                                      Checksumme
0000  C3 38 20 C3 6B 05 C3 C0 05 C3 E4 05 C3 EB 05 C3    += 07F8
0010  D8 04 C3 F9 04 C3 46 05 C3 55 02 C3 65 02 C3 E0    += 0791
0020  06 C3 85 07 C3 69 01 C3 41 01 C3 E5 01 C3 EE 01    += 06E2
0030  C3 1F 02 C3 4F 03 00 00 C3 03 40 C3 F8 01 C3 6E    += 05EC
0040  05 DB 68 CB 7F C2 4E 01 3E FF B7 C3 4F 01 AF C9    += 0822
0050  CD 69 01 FE 7B D2 61 01 FE 61 DA 61 01 D6 61 C6    += 087C
0060  41 C9 00 00 00 00 C3 00 40 CD 78 01 DB 68 CB 7F    += 05E0
0070  C2 69 01 F5 DB 69 F1 C9 3A 46 40 B7 CA A9 01 CD    += 08D7
0080  2B 03 DA A9 01 CD F8 05 3A 47 40 FE 01 C2 A5 01    += 06A4
0090  3A 46 40 32 47 40 3A 67 40 3C E6 03 32 67 40 CD    += 0525
00A0  33 02 C3 A9 01 3D 32 47 40 3A 48 40 B7 CA DB 01    += 05B7
00B0  CD 2B 03 DA DB 01 CD F8 05 3A 49 40 FE 01 C2 D7    += 07D6
00C0  01 3A 48 40 32 49 40 3A 67 40 EE 01 E6 03 32 67    += 04D0
00D0  40 CD 33 02 C3 DB 01 3D 32 49 40 C9 3E 53 D3 CA    += 06D0
00E0  3E 50 D3 CA C9 DB CA E6 01 28 FA DB CB C9 DB CA    += 0AB6
00F0  E6 02 28 FA 79 D3 CB C9 F5 DB 70 E6 04 28 FA F1    += 0A27
0100  C9 CD F8 01 D3 70 C9 CD F8 01 E6 F0 D3 60 3E 07    += 09AF
0110  D3 71 3E 04 CD 01 02 CD F8 01 3E 03 D3 71 C9 3E    += 06A8
0120  00 CD 07 02 3E 50 CD 07 02 3E A0 CD 07 02 3E F0    += 051C
0130  C3 07 02 3A 68 40 07 07 E6 0C 47 3A 67 40 E6 03    += 04BF
0140  B0 07 07 07 07 CD F8 01 C5 E6 F0 47 DB 60 E6 0F    += 07A4
0150  B0 D3 60 C1 C9 CD F8 01 7C D3 78 7D D3 79 7A D3    += 0A10
0160  7A 7B D3 7B C9 22 39 40 ED 53 3B 40 CD F8 01 DB    += 0803
0170  78 CB 5F 28 02 F6 F0 57 DB 79 5F ED 53 3D 40 2A    += 07A3
0180  39 40 AF ED 52 22 41 40 DB 7A CB 5F 28 02 F6 F0    += 0799
0190  57 DB 7B 5F ED 53 3F 40 2A 3B 40 AF ED 52 22 43    += 06C3
01A0  40 ED 5B 41 40 CB 7C 28 03 CD D5 03 CB 7A 28 05    += 0692
01B0  EB CD D5 03 EB 7C B7 C2 BF 02 7A B7 CA EE 02 2A    += 0946
01C0  39 40 ED 5B 3B 40 E5 D5 2A 41 40 CB 2C CB 1D ED    += 076D
01D0  5B 3D 40 19 E5 2A 43 40 CB 2C CB 1D ED 5B 3F 40    += 0629
01E0  19 EB E1 CD 65 02 D1 E1 CD 65 02 C3 F8 02 2A 41    += 0827
01F0  40 ED 5B 43 40 CD F9 02 C9 06 11 CB 7C 28 05 CD    += 06F4
0200  D5 03 CB C8 CB 7A 28 07 EB CD D5 03 EB CB D0 CD    += 09C2
0210  F8 01 7D D3 75 7B D3 77 78 CD 01 02 C9 3E 00 18    += 06EA
0220  02 3E 01 CD 01 02 3E 02 C3 01 02 DB 70 E6 02 28    += 0472
0230  0C 3A 4A 40 B7 37 C0 3C 32 4A 40 AF C9 AF 32 4A    += 0619
0240  40 37 C9 CD F8 01 D3 73 C9 CD F8 01 D3 72 C9 CD    += 09B6
0250  21 03 3E 00 CD 43 03 3E 00 CD 49 03 11 00 00 06    += 02E3
0260  04 21 00 00 CD 55 02 0E 08 3E 0B CD 01 02 0D C2    += 0347
0270  69 03 21 40 00 19 EB 05 C2 61 03 3E 11 C3 43 03    += 0454
0280  11 68 01 7C B7 FA 9C 03 E5 21 99 03 E3 E5 AF ED    += 084C
0290  52 E1 F8 AF ED 52 C3 8D 03 C3 A2 03 19 7C B7 FA    += 091A
02A0  9C 03 C9 0E 00 7A B7 F2 B2 03 0E 01 2F 57 7B 2F    += 058D
02B0  5F 13 78 A9 32 51 40 CB 44 C2 DD 03 7D 0E 08 21    += 05BB
02C0  00 00 29 17 D2 CA 03 19 CE 00 0D C2 C2 03 6C 67    += 052D
02D0  3A 51 40 B7 C8 7C 2F 67 7D 2F 6F 23 C9 EB C3 D0    += 07E1
02E0  03 00 04 09 0D 12 16 1B 1F 24 28 2C 31 35 3A 3E    += 01D5
02F0  42 47 4B 4F 53 58 5C 60 64 68 6C 70 74 78 7C 80    += 061A
0300  84 88 8B 8F 93 96 9A 9E A1 A5 A8 AB AF B2 B5 B8    += 09EE
```

zu Abb. 8.2.11

```
-------------- Seite  2 ------- Datei Grund 2.0 -----------
Rom                                                          Checksumme
0310  BB BE C1 C4 C7 CA CC CF D2 D4 D7 D9 DB DE E0 E2       += 0CFB
0320  E4 E6 E8 EA EC ED EF F1 F2 F3 F5 F6 F7 F8 F9 FA       += 0F07
0330  FB FC FD FE FE FF FF FF 00 00 00 00 EB 21 5A 00       += 0953
0340  AF ED 52 CB 7C CA 4C 04 11 68 01 19 22 56 40 06       += 05A0
0350  00 11 5A 00 AF ED 52 FA 94 04 11 5A 00 AF ED 52       += 0644
0360  FA 87 04 06 01 11 5A 00 AF ED 52 FA 7B 04 21 68       += 05E7
0370  01 ED 5B 56 40 AF ED 52 C3 84 04 2A 56 40 11 B4       += 069D
0380  00 AF ED 52 C3 91 04 ED 5B 56 40 21 B4 00 AF ED       += 0795
0390  52 C3 97 04 2A 56 40 7C B5 C8 EB 21 E1 03 19 6E       += 06E0
03A0  26 00 7D B7 C0 26 01 C9 CD F8 01 DB 78 32 53 40       += 06E8
03B0  DB 79 32 52 40 DB 7A 32 55 40 DB 7B 32 54 40 C9       += 0719
03C0  CD F8 01 3A 53 40 D3 78 3A 52 40 D3 79 3A 55 40       += 06C5
03D0  D3 7A 3A 54 40 D3 7B C9 ED 53 0C 40 E3 E5 11 35       += 07CC
03E0  40 AF ED 52 7C B5 E1 28 06 E5 ED 5B 0C 40 E9 ED       += 08BD
03F0  5B 06 40 D5 ED 5B 0C 40 E9 ED 53 0C 40 ED 43 0A       += 06B9
0400  40 E1 D1 C1 0B 79 B0 20 09 ED 5B 0C 40 ED 4B 0A       += 06E6
0410  40 E9 1B 1B 1B D5 D5 E5 11 35 40 AF ED 52 7C B5       += 07AE
0420  E1 D1 28 0C AF 69 60 ED 5B 0C 40 ED 4B 0A 40 C9       += 073D
0430  D1 ED 53 06 40 E5 69 60 AF C9 21 00 01 11 00 01       += 05B1
0440  01 5A 00 C3 46 05 CD F3 05 22 59 40 EB CD F3 05       += 0699
0450  22 5B 40 69 60 CD 80 03 22 5D 40 CD FD 05 3E 01       += 05A3
0460  32 48 40 32 49 40 AF 32 58 40 C9 CD F3 05 F5 C5       += 0736
0470  D5 E5 E5 3A 58 40 FE 01 C2 7E 05 CD 3A 05 2A 5D       += 0748
0480  40 CD 3C 04 D1 D5 CD A3 03 ED 5B 59 40 19 22 59       += 06DB
0490  40 2A 5D 40 CD 4C 04 D1 CD A3 03 ED 5B 5B 40 19       += 0664
04A0  22 5B 40 2A 59 40 ED 5B 5B 40 CD B9 06 3A 65 40       += 05CE
04B0  FE 01 C2 BB 05 CD 65 02 C3 BE 05 CD 55 02 18 1C       += 0693
04C0  F5 C5 D5 E5 3A 58 40 FE 01 C2 D1 05 E5 CD 3A 05       += 08CE
04D0  E1 ED 5B 5D 40 19 CD 80 03 22 5D 40 CD 78 01 E1       += 0715
04E0  D1 C1 F1 C9 F5 AF 32 65 40 F1 C9 F5 3E 01 32 65       += 094C
04F0  40 F1 C9 29 29 29 29 C9 3A 58 40 B7 C0 E5 D5 C5       += 082F
0500  2A 5F 40 ED 5B 61 40 CD B9 06 CD 55 02 3E 01 32       += 05D3
0510  68 40 CD 33 02 CD 21 03 CD 46 06 2A 5D 40 22 63       += 0500
0520  40 2A 59 40 22 5F 40 ED 5B 5B 40 ED 53 61 40 CD       += 0655
0530  B9 06 CD 55 02 CD 1D 03 CD 46 06 AF 32 68 40 CD       += 063F
0540  33 02 C1 D1 E1 C9 2A 63 40 AF 06 10 CB 25 CB 14       += 06D2
0550  17 FE 2D DA 5A 06 CB C5 D6 2D 10 F0 26 00 7D E6       += 0798
0560  07 6F 29 29 29 11 79 06 19 1E 08 7E B7 CA 73 06       += 0438
0570  CD 01 02 23 1D C2 6B 06 C9 FA FD FF FA 00 00 00       += 06FC
0580  00 D3 D0 D0 D4 D4 D3 00 00 FE F9 FD FE 00 00 00       += 08E0
0590  00 D7 D2 D2 D0 D0 D7 00 00 FC FB F9 FC 00 00 00       += 08DE
05A0  00 D5 D6 D6 D2 D2 D5 00 00 F8 FF FB F8 00 00 00       += 08E4
05B0  00 D1 D4 D4 D6 D6 D1 00 00 C5 06 04 CB 2C CB 1D       += 07A4
05C0  05 C2 BC 06 3A 66 40 FE 01 C2 D1 06 06 05 C3 D3       += 06A2
05D0  06 06 04 CB 2A CB 1B 05 C2 D3 06 C1 C9 CD 1D 03       += 0602
05E0  E5 DD E1 CD 26 09 7E B7 CA 12 07 CD A8 04 CD F8       += 08F5
05F0  01 DB 71 EE 02 D3 71 3E 0A CD 01 02 CD C0 04 CD       += 06F7
0600  F8 01 DB 71 EE 02 D3 71 7E E6 7F CD 01 02 23 C3       += 0812
0610  E6 06 C9 22 7A 40 ED 53 7C 40 32 7E 40 3E 00 32       += 05ED
```

zu Abb. 8.2.11

```
-------------- Seite  3 ------- Datei Grund 2.0 -----------
Rom                                                         Checksumme
0620  7F 40 E5 21 71 07 E3 FD 21 80 40 E5 21 42 07 E3       += 0730
0630  DD 7E 00 DD 23 B7 C8 FE 0A C8 FD 77 00 FD 23 C3       += 0901
0640  30 07 FD 36 00 00 F5 DD E5 21 7A 40 CD E0 06 DD       += 078C
0650  E1 F1 B7 C8 3A 7E 40 E6 0F 4F AF 06 0A 81 05 C2       += 0794
0660  5D 07 4F 06 00 2A 7C 40 AF ED 42 22 7C 40 C3 27       += 0545
0670  07 C9 22 7A 40 ED 53 7C 40 32 7E 40 AF 32 7F 40       += 0638
0680  DD 21 80 40 C9 E5 DD E1 AF DD 77 07 DD 7E 04 CD       += 0960
0690  43 03 DD 7E 05 E6 04 CD 49 03 C5 79 FE 01 C2 A7       += 074F
06A0  07 CD 1D 03 CD A8 08 DD E5 E1 CD 26 09 CD A8 04       += 0789
06B0  CD 21 03 DD 46 06 3E 0A CD 01 02 05 C2 B6 07 DD       += 0593
06C0  E5 E1 11 08 00 19 22 4D 40 36 00 3E 00 32 4B 40       += 03D8
06D0  E5 21 8B 08 E3 CD 41 01 FE FF C2 5C 08 CD 69 01       += 07E5
06E0  E6 7F 32 4C 40 FE 08 CA EF 07 FE 7F C2 2A 08 CD       += 0827
06F0  9C 08 DD 7E 07 B7 CA 27 08 3D DD 77 07 2A 4D 40       += 0605
0700  2B 36 00 22 4D 40 DD 7E 04 0F 0F 0F 0F E6 0F 4F       += 03EF
0710  AF 06 06 81 05 C2 13 08 2A 52 40 4F 06 00 AF ED       += 04CB
0720  42 22 52 40 CD 9C 08 C3 5C 08 FE 20 D2 36 08 CD       += 0689
0730  9C 08 C9 C3 5C 08 F5 2A 4D 40 77 23 36 00 22 4D       += 057F
0740  40 CD 9C 08 CD C0 04 CD 1D 03 F1 CD 01 02 CD A8       += 0765
0750  04 DD 7E 07 3C DD 77 07 DD BE 06 C8 CD 2B 03 DA       += 073B
0760  88 08 3A 4B 40 3C 32 4B 40 FE 0D C2 7C 08 CD C0       += 062C
0770  04 CD 1D 03 3E 0A CD 01 02 C3 88 08 FE 1A C2 88       += 05BE
0780  08 CD 9C 08 AF 32 4B 40 C3 D5 07 C1 79 FE 01 C2       += 077F
0790  98 08 CD 21 03 CD A8 08 3A 4C 40 C9 CD C0 04 CD       += 06FB
07A0  21 03 3E 0A CD 01 02 C9 DD E5 E1 CD 26 09 3E D7       += 06B9
07B0  CD 01 02 DD 7E 04 0F 0F 0F 0F E6 0F 4F AF 06 06       += 046A
07C0  81 05 C2 C0 08 4F 06 00 21 00 00 DD 7E 06 09 3D       += 042D
07D0  C2 CE 08 23 23 E5 DD 7E 04 E6 0F 4F AF 06 08 81       += 06A4
07E0  05 C2 DF 08 3C 3C 3C CD F8 01 D3 77 E1 CB 24 24       += 0766
07F0  7D D3 75 3E 12 CD 01 02 44 3E 10 CD 01 02 CD F8       += 060C
0800  01 3E 80 D3 75 05 C2 F9 08 CD F8 01 7D D3 75 3E       += 0798
0810  14 CD 01 02 44 3E 16 CD 01 02 CD F8 01 3E 80 D3       += 05A3
0820  75 05 C2 15 09 C9 CD F8 01 7E D3 79 23 7E D3 78       += 079F
0830  23 7E D3 7B 23 7E D3 7A 23 7E D3 73 23 7E D3 72       += 07AA
0840  23 C9 22 D1 40 ED 53 D3 40 32 D5 40 3E 00 32 D6       += 06FF
0850  40 78 32 D7 40 0E 01 21 D1 40 CD 85 07 C9 7C CD       += 06AD
0860  63 09 7D F5 0F 0F 0F 0F E6 0F CD 70 09 F1 E6 0F       += 063B
0870  FE 0A DA 7C 09 D6 0A C6 41 C3 7E 09 C6 30 DD 77       += 07E2
0880  00 DD 36 01 00 DD 23 C9 C5 06 08 07 D2 96 09 DD       += 0605
0890  36 00 31 C3 9A 09 DD 36 00 30 DD 23 05 C2 8B 09       += 056B
08A0  DD 36 00 00 C1 C9 7E B7 CA B5 09 7E DD 77 00 DD       += 0809
08B0  23 23 C3 A6 09 DD 36 00 00 C9 DD 36 00 20 DD 36       += 05DA
08C0  01 00 DD 23 C9 DD 77 00 DD 36 01 00 DD 23 C9 DD       += 06D8
08D0  E5 CD D4 0A DD 7E 00 FE 7B 30 05 FE 61 D2 EA 09       += 08BD
08E0  FE 5B D2 41 0A FE 41 DA 41 0A AF DA F9 09 DD 23       += 0865
08F0  DD 7E 00 CD 49 0C C3 EB 09 CD D4 0A DD 7E 00 FE       += 0838
0900  3A C2 41 0A DD 23 DD 7E 00 FE 3D C2 41 0A DD 23       += 06EA
0910  CD D4 0A FE 25 C2 27 0A DD 23 DD E3 CD 70 0C DD       += 08A7
0920  E1 D8 AF C9 C3 41 0A CD 48 0A DA 41 0A DD E3 E5       += 0928
```

zu Abb. 8.2.11

```
-------------- Seite  4  ------- Datei Grund 2.0 -----------
Rom                                                     Checksumme
0930  DD E5 CD 70 0C DD E1 E1 E5 CD EE 0C E1 DD E1 AF    += 0BA4
0940  C9 DD E1 21 00 00 37 C9 CD F2 0A D8 DD 7E 00 FE    += 08A2
0950  2B CA 59 0A FE 2D C2 7E 0A DD 23 FE 2B C2 6A 0A    += 072C
0960  E5 CD F2 0A D1 D8 19 C3 74 0A E5 CD F2 0A D1 EB    += 0A1B
0970  D8 AF ED 52 AF 32 69 40 DD 7E 00 C3 4F 0A 3A 69    += 076A
0980  40 E6 FE B7 C2 99 0A 7C B7 CA 91 0A FE FF C2 99    += 0A30
0990  0A 3A 69 40 F6 80 32 69 40 DD 7E 00 FE 2E C2 CF    += 0756
09A0  0A DD 23 DD 7E 00 CD 43 0D FE 57 C2 BB 0A DD 23    += 075E
09B0  3A 69 40 CB BF 32 69 40 C3 CF 0A FE 42 C2 CD 0A    += 07BD
09C0  DD 23 3A 69 40 CB FF 32 69 40 C3 CF 0A DD 2B AF    += 07DB
09D0  3A 69 40 C9 DD 7E 00 FE 20 C2 E4 0A DD 23 DD 7E    += 0830
09E0  00 C3 D7 0A C9 CD D4 0A B7 C2 F0 0A AF C3 F1 0A    += 08F8
09F0  37 C9 DD 7E 00 FE 2D C2 0D 0B DD 23 CD F2 0A EB    += 0814
0A00  21 00 00 AF ED 52 AF 32 69 40 C3 44 0B 3E 01 32    += 051C
0A10  69 40 FD 21 C9 41 CD D0 0B D2 44 0B 3E 02 32 69    += 0675
0A20  40 FD 21 45 0B CD D0 0B D2 44 0B 3E 00 32 69 40    += 0590
0A30  DD 7E 00 FE 24 C2 41 0B DD 23 2A 6E 40 AF C3 44    += 0719
0A40  0B CD 52 0D C9 53 43 48 52 45 49 54 C5 03 01 53    += 052E
0A50  43 48 52 31 36 54 45 CC 3E 01 44 52 45 48 C5 06    += 04D6
0A60  01 48 45 42 C5 09 01 53 45 4E 4B C5 0C 01 53 43    += 0438
0A70  48 4C 45 49 46 C5 0F 01 45 4E 44 53 43 48 4C 45    += 0483
0A80  49 46 C5 12 01 53 45 D4 15 01 4D 4F 56 45 54 CF    += 0543
0A90  18 01 44 52 41 57 54 CF 1B 01 57 52 49 54 C5 1E    += 04AF
0AA0  01 52 45 41 C4 21 01 43 C9 24 01 43 53 54 D3 27    += 04D4
0AB0  01 52 C9 2A 01 50 CF 2D 01 43 4C D2 30 01 43 4C    += 04B5
0AC0  50 C7 33 01 57 41 49 D4 3B 01 52 41 CD 00 48 00    += 04E4
0AD0  DD 7E 00 CD 49 0C D8 E5 21 48 0C E3 FD 7E 00 B7    += 07C4
0AE0  C2 E5 0B 37 C9 FD 22 74 40 DD 22 76 40 FD 46 00    += 077D
0AF0  CB B8 DD 7E 00 CD 43 0D B8 C2 23 0C DD 23 FD CB    += 086C
0B00  00 7E CA 1D 0C DD 7E 00 CD 49 0C D2 19 0C FD 6E    += 0650
0B10  01 FD 66 02 AF C9 C3 1A 0C 37 C3 20 0C FD 23 AF    += 06BC
0B20  C3 24 0C 37 D2 ED 0B DD 2A 76 40 FD 2A 74 40 FD    += 0789
0B30  CB 00 7E C2 3F 0C FD 23 FD CB 00 7E C3 33 0C FD    += 07BB
0B40  23 FD 23 FD 23 C3 DC 0B C9 FE 3A 30 05 FE 30 D2    += 0843
0B50  69 0C FE 5B 30 05 FE 41 D2 69 0C FE 7B 30 05 FE    += 0735
0B60  61 D2 69 0C FE 5F C2 6E 0C 37 3F C3 6F 0C 37 C9    += 06F5
0B70  FD 21 C9 41 CD D0 0B D8 E5 2A C1 41 2B 22 C1 41    += 0808
0B80  2A 74 40 11 00 00 CB 7E C2 92 0C 13 23 CB 7E C3    += 05DA
0B90  88 0C 23 13 23 13 23 13 E5 2A C3 41 C1 C5 AF ED    += 066B
0BA0  42 44 4D 03 2A C3 41 AF ED 52 22 C3 41 E1 ED 5B    += 0741
0BB0  74 40 ED B0 E1 C9 EB E5 21 E9 0C E3 FD 22 78 40    += 099B
0BC0  FD 7E 00 B7 C2 C9 0C 37 C9 FD CB 00 7E FD 23 CA    += 08F9
0BD0  C9 0C FD 6E 00 FD 23 FD 66 00 FD 23 AF ED 52 7D    += 084E
0BE0  B4 C2 E6 0C AF C9 C3 BC 0C FD 2A 78 40 C9 FD 21    += 0931
0BF0  C9 41 E5 DD E5 CD D0 0B DD E1 E1 DA 00 0D 37 C9    += 09DF
0C00  DD 7E 00 CD 49 0C D8 FD 2A C3 41 DD 7E 00 CD 43    += 07EB
0C10  0D FD 77 00 DD 23 FD 23 DD 7E 00 CD 49 0C D2 0B    += 06FB
0C20  0D FD 2B FD CB 00 FE FD 75 01 FD 74 02 FD 36 03    += 0817
0C30  00 11 03 00 FD 19 FD 22 C3 41 2A C1 41 23 22 C1    += 057F
```

zu Abb. 8.2.11

```
-------------- Seite  5  ------- Datei Grund 2.0 -----------
Rom                                                           Checksumme
0C40  41 AF C9 FE 7B D2 51 0D FE 61 DA 51 0D D6 61 C6         += 08F6
0C50  41 C9 DD 7E 00 FE 2D C2 61 0D DD 23 3E 01 C3 6D         += 072F
0C60  0D FE 2B C2 6C 0D DD 23 AF C3 6D 0D AF 32 45 40         += 06C3
0C70  21 00 00 DD 7E 00 FE 23 CA C0 0D DD 7E 00 CD 07         += 0663
0C80  0E D8 DA BC 0D DD 23 29 29 29 29 FE 47 D2 9C 0D         += 06ED
0C90  FE 41 DA 9C 0D D6 41 C6 0A C3 AF 0D FE 67 D2 AD         += 090C
0CA0  0D FE 61 DA AD 0D D6 61 C6 0A C3 AF 0D D6 30 4F         += 07DB
0CB0  06 00 09 DD 7E 00 CD 07 0E C3 82 0D AF C3 E4 0D         += 0601
0CC0  DD 23 DD 7E 00 CD F6 0D D8 DA E3 0D DD 23 54 5D         += 087E
0CD0  29 29 19 29 D6 30 4F 06 00 09 DD 7E 00 CD F6 0D         += 0523
0CE0  C3 C9 0D AF F5 3A 45 40 FE 01 C2 F4 0D EB 21 00         += 07CA
0CF0  00 AF ED 52 F1 C9 FE 3A D2 05 0E FE 30 DA 05 0E         += 07E0
0D00  37 3F C3 06 0E 37 C9 FE 47 30 05 FE 41 D2 23 0E         += 0609
0D10  FE 67 30 05 FE 61 D2 23 0E FE 3A D2 28 0E FE 30         += 076A
0D20  DA 28 0E 37 3F C3 29 0E 37 C9 06 00 54 5D 7E E6         += 059B
0D30  DF FE DD CA A0 0E 7E FE CB CA 9A 0E FE ED CA 8E         += 0B2E
0D40  0E 7E FE C3 CA DC 0E FE CD CA DC 0E E6 EF FE 22         += 0A75
0D50  CA DC 0E FE 2A CA DC 0E E6 CF FE 01 CA DC 0E E6         += 09DE
0D60  C7 FE C2 CA DC 0E FE C4 CA DC 0E 7E E6 F7 FE 10         += 0B1A
0D70  CA DD 0E FE D3 CA DD 0E E6 E7 FE 20 CA DD 0E E6         += 0AC1
0D80  C7 FE 06 CA DD 0E FE C6 CA DD 0E C3 DE 0E 23 7E         += 0949
0D90  E6 C7 FE 43 CA DB 0E C3 DD 0E C3 DD 0E C3 DB 0E         += 09A9
0DA0  23 7E FE CB CA DB 0E FE 21 CA DB 0E E6 FE FE 34         += 0A05
0DB0  CA DC 0E E6 F8 FE 70 CA DC 0E 7E FE 36 CA DB 0E         += 0A19
0DC0  E6 C7 FE 06 CA DC 0E E6 C7 FE 02 CA DB 0E 7E D6         += 0A19
0DD0  40 E6 87 FE 06 CA DC 0E C3 DD 0E 04 04 04 04 EB         += 070E
0DE0  C9 ED 43 0A 40 ED 53 0C 40 22 0E 40 F5 C1 ED 43         += 0725
0DF0  08 40 08 D9 ED 43 12 40 ED 53 14 40 22 16 40 F5         += 05AC
0E00  C1 ED 43 10 40 ED 57 32 1E 40 ED 5F 32 1F 40 DD         += 06CF
0E10  22 18 40 FD 22 1A 40 2A 06 40 C9 22 06 40 FD 2A         += 04BB
0E20  1A 40 DD 2A 18 40 3A 1E 40 ED 47 ED 4B 10 40 C5         += 05D2
0E30  F1 2A 16 40 ED 5B 14 40 ED 4B 12 40 D9 08 ED 4B         += 06B0
0E40  08 40 C5 F1 2A 0E 40 ED 5B 0C 40 ED 4B 0A 40 C9         += 0655
0E50  E5 C5 CD F8 01 0E 70 21 20 40 06 10 ED 78 77 0C         += 066D
0E60  23 05 C2 5C 0F C1 E1 C9 E5 C5 CD F8 01 0E 71 21         += 07D0
0E70  21 40 06 0F 7E ED 79 0C 23 05 C2 74 0F C1 E1 C9         += 063E
0E80  3E C3 32 36 40 E5 21 B3 0F 22 37 40 E1 AF 06 04         += 05A4
0E90  11 35 40 12 1B 10 FC D5 CD 2A 0E D1 7E 13 12 23         += 0530
0EA0  10 FA CD 68 0F CD 1B 0F ED 73 30 40 ED 7B 1C 40         += 06D9
0EB0  C3 32 40 ED 73 1C 40 ED 7B 30 40 CD E1 0E CD 50         += 07A2
0EC0  0F C9 7E 23 ED 73 30 40 ED 7B 1C 40 E5 ED 73 1C         += 076E
0ED0  40 ED 7B 30 40 26 00 E6 38 6F C9 23 23 23 ED 73         += 065D
0EE0  30 40 ED 7B 1C 40 E5 ED 73 1C 40 ED 7B 30 40 2B         += 06D8
0EF0  7E 2B 6E 67 C9 2A 0E 40 C9 2A 18 40 C9 2A 1A 40         += 0557
0F00  C9 ED 73 30 40 ED 7B 1C 40 E1 ED 73 1C 40 ED 7B         += 0862
0F10  30 40 C9 23 7E 23 66 6F C9 23 7E 23 4F B7 FA 25         += 0684
0F20  10 06 00 09 C9 06 FF 09 C9 ED 4B 0A 40 05 ED 43         += 0576
0F30  0A 40 20 E5 23 23 C9 E5 21 DB 0F 22 33 40 21 5B         += 055F
0F40  10 22 36 40 3E C3 32 35 40 E1 7E E6 38 F6 C2 32         += 06B7
```

zu Abb. 8.2.11

```
-------------- Seite  6 ------- Datei Grund 2.0 -----------
Rom                                                          Checksumme
0F50  32 40 ED 4B 08 40 C5 F1 C3 32 40 23 23 23 C9 E5     += 06F4
0F60  21 13 10 22 33 40 21 5B 10 22 36 40 3E C3 32 35     += 0365
0F70  40 E1 7E 32 32 40 ED 4B 08 40 C5 F1 C3 32 40 E5     += 0793
0F80  21 01 10 22 33 40 21 5D 10 22 36 40 3E C3 32 35     += 0355
0F90  40 E1 7E E6 38 F6 C2 32 32 40 ED 4B 08 40 C5 F1     += 084F
0FA0  C3 32 40 E5 21 19 10 22 33 40 21 5C 10 22 36 40     += 041E
0FB0  3E C3 32 35 40 E1 7E E6 18 F6 C2 32 32 40 ED 4B     += 0799
0FC0  08 40 C5 F1 C3 32 40 7E FE C9 CA 01 10 FE CD CA     += 08E8
0FD0  3D 11 FE C3 CA 13 10 FE 18 CA 19 10 FE 10 CA 29     += 0706
0FE0  10 FE E9 CA F5 0F E6 E7 FE 20 CA A3 10 E6 C7 FE     += 0AD8
0FF0  C2 CA 5F 10 FE C4 CA 32 11 FE C7 CA C2 0F FE C0     += 09E8
1000  CA 7F 10 7E FE ED C2 19 11 23 7E 2B FE 4D CA 01     += 0790
1010  10 FE 45 CA 01 10 C3 80 0F 7E E6 DF FE DD C2 80     += 08E0
1020  0F 23 7E 2B FE E9 C2 80 0F 7E FE DD CA F9 0F C3     += 0901
1030  FD 0F 06 03 CD 06 18 D2 80 0F C3 37 10 06 03 CD     += 0541
1040  06 18 D2 80 0F C3 DB 0F 50 00 DC 00 33 00 45 50     += 0520
1050  52 4F 4D 20 50 52 4F 47 00 50 00 DC 00 33 00 45     += 03EA
1060  50 52 4F 4D 20 6C 65 73 65 6E 00 76 6F 6E 20 0A     += 04F2
1070  62 69 73 20 0A 6E 61 63 68 00 50 00 50 00 22 00     += 03C4
1080  42 65 72 65 69 74 20 3D 20 42 00 50 00 32 00 22     += 03BE
1090  00 50 72 6F 6D 20 6F 6B 20 4D 3D 4D 65 6E 00 50     += 04B2
10A0  00 32 00 22 00 50 52 4F 4D 20 46 45 48 4C 45 52     += 0368
10B0  20 4D 3D 4D 65 6E 00 3E 40 D3 82 21 59 11 CD DD     += 05D2
10C0  06 DD 21 6B 11 CD 54 14 D8 DD 21 D9 40 CD 48 0A     += 06C3
10D0  D8 22 72 40 2A 6E 40 ED 5B 70 40 DD 2A 72 40 7D     += 06B2
10E0  D3 81 7C E6 1F F6 40 D3 82 DB 80 DD 77 00 E5 AF     += 09A3
10F0  ED 52 7D B4 E1 C8 DD 23 23 18 E4 21 48 11 CD DD     += 085C
1100  06 DD 21 6B 11 CD 54 14 D8 DD 21 D9 40 CD 48 0A     += 06C3
1110  D8 22 72 40 21 7A 11 CD DD 06 3E 40 D3 82 CD 50     += 06F8
1120  01 FE 42 C0 2A 6E 40 ED 5B 70 40 ED 4B 72 40 3E     += 06F9
1130  80 D3 82 7E D3 80 79 D3 81 78 E6 1F F6 80 D3 82     += 09BB
1140  F6 20 D3 82 E6 DF D3 82 E5 D5 C5 C5 CD 2B 03 DA     += 0A9E
1150  4C 12 21 FA 00 11 50 00 3E 33 CD 72 07 E1 CD 5E     += 059D
1160  09 21 7A 40 CD DD 06 C1 D1 E1 CD 2B 03 DA 6A 12     += 0758
1170  CD 2B 03 DA 70 12 E5 AF ED 52 7D B4 E1 28 04 23     += 078B
1180  03 18 B0 3E 40 D3 82 2A 6E 40 ED 5B 70 40 ED 4B     += 06A6
1190  72 40 79 D3 81 78 E6 1F F6 40 D3 82 DB 80 BE 20     += 08C0
11A0  0D E5 AF ED 52 7D B4 E1 28 09 23 03 18 E4 21 9F     += 0705
11B0  11 18 03 21 8B 11 CD 41 01 28 03 CD 50 01 CD DD     += 04EB
11C0  06 CD 50 01 FE 4D C2 C1 12 C9 AF 21 EF 13 18 05     += 06BC
11D0  21 03 14 3E 01 32 4F 40 CD DD 06 21 14 00 11 80     += 03AE
11E0  00 3E 22 CD 72 07 CD E5 01 FE FF C2 E6 12 CD E5     += 08C2
11F0  01 B7 C2 EE 12 CD E5 01 FE 2F C2 E6 12 CD E5 01     += 08C7
1200  FE 0D CA 10 13 DD 77 00 DD 23 CD E5 01 C3 00 13     += 06D5
1210  DD 36 00 00 21 7A 40 CD DD 06 0E 3A CD 33 13 DA     += 05D3
1220  27 13 0E 24 CD 33 13 CD DD 06 CD 50 01 FE 4D C2     += 065A
1230  2A 13 C9 CD E5 01 B7 C2 33 13 CD E5 01 B9 C2 33     += 07D9
1240  13 21 00 00 CD E5 01 57 CD E5 01 5F ED 53 6E 40     += 063E
1250  19 D5 CD E5 01 57 CD E5 01 5F ED 53 70 40 19 D1     += 07E4
```

zu Abb. 8.2.11

```
-------------- Seite  7  ------- Datei Grund 2.0 -----------
Rom                                                          Checksumme
1260  E5 21 8F 13 E3 CD E5 01 4F 06 00 09 3A 4F 40 B7       += 061C
1270  C2 75 13 79 12 1A B9 CA 7C 13 37 C9 E5 2A 70 40       += 06C0
1280  AF ED 52 7C B5 E1 C2 8B 13 AF C9 13 C3 65 13 EB       += 0911
1290  21 B1 13 30 04 21 D7 13 C9 CD E5 01 BA 28 05 21       += 05A8
12A0  C0 13 37 C9 CD E5 01 BB 28 05 21 C0 13 37 C9 AF       += 0711
12B0  C9 50 00 32 00 22 00 4F 4B 20 4D 3D 4D 65 6E 00       += 03D1
12C0  50 00 32 00 22 00 46 65 68 6C 65 72 20 43 41 53       += 03F1
12D0  20 4D 3D 4D 65 6E 00 50 00 32 00 22 00 46 65 68       += 0381
12E0  6C 65 72 20 56 67 6C 2E 20 4D 3D 4D 65 6E 00 50       += 04D4
12F0  00 DC 00 33 00 4C 61 64 65 6E 20 76 6F 6E 20 43       += 04C9
1300  41 53 00 50 00 DC 00 33 00 50 72 75 65 66 6C 65       += 04C6
1310  73 65 6E 20 43 41 53 00 50 00 DC 00 33 00 61 75       += 0472
1320  66 20 43 41 53 20 73 70 65 69 63 68 65 72 6E 00       += 053E
1330  53 74 61 72 74 61 64 72 2E 20 3A 0A 45 6E 64 61       += 054F
1340  64 72 2E 20 20 20 3A 0A 4E 61 6D 65 20 20 20 20       += 03A9
1350  20 20 3A 00 21 0A 00 11 B4 00 3E 22 CD 13 07 21       += 02D2
1360  96 00 11 B4 00 3E 22 06 1E CD 42 09 DD 21 D9 40       += 050E
1370  CD 48 0A 22 6E 40 D8 21 96 00 11 A0 00 3E 22 06       += 0495
1380  1E CD 42 09 DD 21 D9 40 CD 48 0A 22 70 40 D8 21       += 0637
1390  96 00 11 8C 00 3E 22 06 1E CD 42 09 AF C9 21 18       += 0480
13A0  14 CD DD 06 DD 21 30 14 CD 54 14 D8 06 28 0E FF       += 064E
13B0  CD EE 01 05 C2 AE 14 0E 00 CD EE 01 0E 2F CD EE       += 0707
13C0  01 DD 21 D9 40 DD 7E 00 B7 CA D7 14 DD 4E 00 DD       += 07E7
13D0  23 CD EE 01 C3 C5 14 0E 0D CD EE 01 0E 3A CD FE       += 0765
13E0  14 21 C1 41 22 6E 40 2A C3 41 22 70 40 0E 24 CD       += 0506
13F0  FE 14 06 14 0E FF CD EE 01 05 C2 F4 14 C9 C5 06       += 0758
1400  14 0E FF CD EE 01 05 C2 01 15 0E 00 CD EE 01 C1       += 0645
1410  CD EE 01 21 00 00 ED 5B 6E 40 19 D5 4A CD EE 01       += 06C7
1420  4B CD EE 01 ED 5B 70 40 19 4A CD EE 01 4B CD EE       += 0824
1430  01 D1 E5 21 4E 15 E3 1A 4F 06 00 09 CD EE 01 E5       += 0637
1440  2A 70 40 AF ED 52 7C B5 E1 C8 13 C3 37 15 4C CD       += 07DD
1450  EE 01 4D CD EE 01 C9 50 00 DC 00 33 00 49 4F 20       += 05D8
1460  6C 65 73 65 6E 00 0A 00 0A 00 22 00 52 3D 41 64       += 0381
1470  72 20 44 3D 44 61 75 65 72 20 53 3D 53 74 6F 70       += 055A
1480  20 4D 3D 4D 65 6E 75 65 00 21 57 15 CD DD 06 CD       += 05AE
1490  1D 03 21 66 15 CD DD 06 CD 6B 1C D8 4D C5 CD D8       += 074F
14A0  15 CD 50 01 C1 FE 44 C2 CF 15 C5 CD 2B 03 DA B4       += 082A
14B0  15 CD D8 15 CD 41 01 FE FF C2 BF 15 CD 50 01 C1       += 0850
14C0  FE 53 CA CF 15 FE 4D CA CF 15 FE 41 C2 AA 15 FE       += 09B6
14D0  4D C8 FE 52 28 B3 18 C5 C5 21 C8 00 11 96 00 3E       += 06B0
14E0  22 CD 72 07 C1 ED 78 F5 CD 63 09 21 7A 40 CD DD       += 0841
14F0  06 21 96 00 11 64 00 3E 22 CD 72 07 F1 CD 88 09       += 0527
1500  21 7A 40 CD DD 06 C9 50 00 DC 00 33 00 49 4F 20       += 056B
1510  73 65 74 7A 65 6E 00 14 00 96 00 22 00 44 61 74       += 047E
1520  61 3A 20 20 20 00 0A 00 14 00 22 00 4D 3D 4D 65       += 0277
1530  6E 75 65 20 52 3D 41 64 72 00 21 07 16 CD DD 06       += 04FC
1540  CD 6B 1C D8 4D C5 21 17 16 CD DD 06 21 5A 00 11       += 05C8
1550  96 00 3E 22 06 1E CD 42 09 DD 21 D9 40 CD 48 0A       += 0568
1560  C1 D8 ED 69 21 26 16 CD DD 06 CD 50 01 FE 52 28       += 0792
```

zu Abb. 8.2.11

```
------------- Seite  8  ------- Datei Grund 2.0 -----------
Rom                                                     Checksumme
1570  C9 FE 4D C8 18 F4 50 00 DC 00 33 00 53 79 6D 62    += 06E2
1580  6F 6C 65 20 61 75 73 67 65 62 65 6E 00 14 00 BE    += 057C
1590  00 22 00 2B 3D 77 65 69 74 65 72 20 4E 3D 4E 6F    += 0482
15A0  63 68 6D 61 6C 20 4D 3D 4D 65 6E 75 65 00 FD 21    += 05C7
15B0  C9 41 FD E5 CD 1F 02 21 76 16 CD DD 06 21 8D 16    += 06FB
15C0  CD DD 06 FD E1 06 08 21 0A 00 CD EA 16 21 04 01    += 05BA
15D0  06 08 CD EA 16 CD 50 01 E6 7F FE 4D CA E4 16 FE    += 086B
15E0  4E C2 B2 16 FE 4D C2 AE 16 C9 22 7A 40 E5 21 75    += 07C9
15F0  17 E3 21 EE FF 11 14 00 C5 19 05 C2 F9 16 C1 22    += 06C4
1600  7C 40 3E 22 32 7E 40 3E 00 32 7F 40 FD 7E 00 B7    += 056D
1610  C8 DD 21 86 40 1E 0E FD 7E 00 DD 77 00 DD CB 00    += 072F
1620  BE 1D CA 2E 17 DD 23 FD 23 CB 7F C3 42 17 FD CB    += 0838
1630  00 7E FD 23 DD 23 C2 42 17 FD CB 00 7E FD 23 C3    += 07E2
1640  36 17 CA 17 17 DD 36 00 00 FD 6E 00 FD 23 FD 66    += 0646
1650  00 FD 23 FD E5 C5 DD 21 80 40 CD 5E 09 DD 36 00    += 07CC
1660  20 DD 36 01 20 21 7A 40 CD DD 06 C1 FD E1 05 78    += 06FB
1670  B7 C8 C3 F2 16 C9 50 00 DC 00 33 00 61 65 6E 64    += 070A
1680  65 72 6E 00 2B 20 20 3D 41 64 72 20 2B 20 31 20    += 03C0
1690  20 2D 20 3D 41 64 72 20 2D 20 31 0A 4D 20 20 3D    += 0333
16A0  4D 65 6E 75 65 20 20 20 20 52 20 3D 41 64 72 0A    += 044A
16B0  63 72 20 3D 65 69 6E 20 42 65 66 65 68 6C 20 77    += 056B
16C0  65 69 74 65 72 00 21 76 17 CD DD 06 21 14 00 11    += 04BD
16D0  32 00 3E 22 DD 21 84 17 CD 13 07 C9 21 5D 1C CD    += 0542
16E0  DD 06 21 44 00 11 BE 00 3E 22 06 1E CD 42 09 DD    += 0490
16F0  21 D9 40 CD 48 0A C9 C5 CD BA 09 7E 23 CD 63 09    += 0751
1700  C1 05 C2 F7 17 C9 E5 D5 58 16 00 1B 1B 19 5E 23    += 0657
1710  56 FD 21 C9 41 EB E5 CD B6 0C E1 D2 25 18 FD 21    += 08EB
1720  45 0B CD B6 0C D1 E1 C9 CD C6 17 CD DC 17 22 6E    += 0854
1730  40 22 6C 40 E5 21 A7 19 E3 2A 6E 40 CD 2A 0E 48    += 05DC
1740  06 00 09 22 70 40 21 14 00 11 A5 00 3E 22 CD 72    += 036B
1750  07 2A 6C 40 ED 5B 6E 40 AF ED 52 7D B4 CA 66 18    += 073A
1760  2A 6C 40 CD C4 19 21 14 00 11 91 00 3E 22 CD 72    += 04F6
1770  07 2A 6E 40 22 6C 40 CD C4 19 21 14 00 11 7D 00    += 041A
1780  3E 22 CD 72 07 2A 70 40 CD C4 19 21 14 00 11 64    += 04D4
1790  00 3E 22 CD 72 07 2A 6E 40 CD 5E 09 21 7A 40 CD    += 055A
17A0  DD 06 21 5C 00 11 64 00 3E 22 06 1E CD 42 09 3A    += 03AB
17B0  D8 40 B7 C2 BB 18 3E 0D C3 CB 18 FE 01 C2 C9 18    += 07F7
17C0  3A D9 40 CD 53 01 C3 CB 18 3E 00 FE 2B C2 DA 18    += 0735
17D0  2A 6E 40 23 22 6E 40 C3 A4 19 FE 2D C2 ED 18 2A    += 0667
17E0  6E 40 2B 22 6E 40 2B 22 6C 40 C3 A4 19 FE 0D C2    += 05EF
17F0  02 19 2A 6E 40 CD 2A 0E 48 06 00 09 22 6E 40 C3    += 03E2
1800  A4 19 FE 4D C2 0B 19 C9 C3 A4 19 FE 52 C2 19 19    += 077B
1810  CD DC 17 22 6E 40 C3 A4 19 DD 21 D9 40 CD D4 0A    += 07D2
1820  DD 7E 00 FE 22 CA 2D 19 FE 27 C2 5C 19 32 50 40    += 06A9
1830  DD 23 FD 2A 6E 40 E5 21 55 19 E3 3A 50 40 DD BE    += 0791
1840  00 C8 DD 7E 00 FE 80 D0 FE 20 D8 FD 77 00 DD 23    += 08DB
1850  FD 23 C3 3B 19 FD 22 6E 40 C3 A4 19 DD 21 D9 40    += 079B
1860  CD CF 09 D2 A4 19 DD 21 D9 40 CD 48 0A FD 2A 6E    += 07FF
1870  40 DA 9A 19 FD 75 00 FD 23 CB 7F C2 83 19 FD 74    += 0878
```

zu Abb. 8.2.11

```
-------------- Seite  9  ------- Datei Grund 2.0 -----------
Rom                                                          Checksumme
1880  00 FD 23 CD D4 0A DD 7E 00 FE 2C C2 90 19 DD 23       += 07BB
1890  FD E5 CD 48 0A FD E1 C3 71 19 CD E5 0A DA A4 19       += 097F
18A0  FD 22 6E 40 C3 39 18 C9 C5 7E E6 DF FE DD 20 01       += 08AE
18B0  05 7E FE ED 20 01 05 78 FE 03 C2 C1 19 37 C3 C2       += 0765
18C0  19 AF C1 C9 CD 5E 09 CD BA 09 3E 3A CD C5 09 CD       += 07F6
18D0  2A 0E E5 C5 CD F7 17 C1 E1 CD A8 19 D2 23 1A CD       += 08C9
18E0  06 18 DA 23 1A CD BA 09 3E 7C CD C5 09 CD BA 09       += 06AA
18F0  7E CD 63 09 CD BA 09 7E FE DD CA 07 1A FE FD CA       += 0950
1900  07 1A FE ED C2 0F 1A 23 7E CD 63 09 CD BA 09 23       += 0684
1910  FD 7E 00 E6 7F DD 77 00 FD CB 00 7E DD 23 FD 23       += 089A
1920  CA 10 1A 06 28 CD BA 09 05 C2 25 1A 21 7A 40 CD       += 0560
1930  DD 06 C9 50 00 DC 00 33 00 45 69 6E 7A 65 6C 73       += 05E5
1940  63 68 72 69 74 74 00 00 00 0A 00 11 00 61 66 20       += 0390
1950  20 20 62 63 20 20 20 64 65 20 20 20 68 6C 20 20       += 03A2
1960  20 61 66 27 20 20 62 63 27 20 20 64 65 27 20 20       += 03AA
1970  68 6C 27 20 20 69 78 20 20 20 69 79 20 20 20 73       += 0431
1980  70 20 20 20 69 20 20 72 00 21 47 1A CD DD 06 21       += 043E
1990  00 00 11 00 00 3E 11 CD 72 07 21 08 40 06 0B C5       += 02E5
19A0  23 7E 2B CD 63 09 7E 23 23 CD 63 09 CD BA 09 C1       += 0653
19B0  05 C2 9F 1A 7E 23 CD 63 09 CD BA 09 7E CD 63 09       += 06A1
19C0  21 7A 40 CD DD 06 C9 3A 68 40 F5 3E 02 F5 3D 32       += 06CF
19D0  68 40 CD 33 02 CD 1D 03 CD E5 1A F1 3D C2 CD 1A       += 073A
19E0  F1 32 68 40 C9 CD 89 1A 21 54 01 11 14 00 3E 11       += 04EE
19F0  CD 72 07 2A 06 40 CD C4 19 21 00 1B CD DD 06 C9       += 0615
1A00  00 00 14 00 11 00 52 3D 41 64 72 20 53 3D 53 65       += 0333
1A10  69 74 65 20 4E 3D 6E 2D 4D 61 6C 20 42 3D 42 69       += 04EC
1A20  73 20 63 72 3D 42 65 66 65 68 6C 20 61 75 73 66       += 05BA
1A30  75 65 68 72 65 6E 20 4D 3D 4D 65 6E 75 65 00 AF       += 05DA
1A40  32 67 40 32 68 40 CD 33 02 21 9C 1C 22 FE 4F 21       += 051E
1A50  FE 4F 22 1C 40 21 33 1A CD DD 06 CD 6B 1C D8 22       += 0637
1A60  06 40 CD 1C 1C CD C7 1A CD 50 01 FE 0D 20 08 CD       += 0617
1A70  8E 1B DA 9C 1C 18 EE FE 52 28 C4 FE 53 28 27 FE       += 081B
1A80  4E 28 46 FE 42 28 57 FE 4D 20 DA C3 D4 1C 2A 06       += 06A3
1A90  40 11 9C 1C AF ED 52 7C B5 37 C8 2A 06 40 CD C7       += 072B
1AA0  10 22 06 40 AF C9 CD FC 1B D2 B7 1B 3E 01 32 48       += 0631
1AB0  40 32 49 40 C3 C7 1B AF 32 48 40 7D E6 03 32 67       += 0608
1AC0  40 32 68 40 CD 33 02 18 9C CD FC 1B 38 97 E5 CD       += 0735
1AD0  8E 1B E1 DA 9C 1C 2B 7D B4 20 F3 C3 65 1B CD FC       += 0897
1AE0  1B DA 65 1B E5 CD 8E 1B E1 DA 9C 1C E5 ED 5B 06       += 0876
1AF0  40 AF ED 52 7C B5 E1 20 EB C3 65 1B 3E 00 32 67       += 0765
1B00  40 32 68 40 CD 33 02 21 05 00 11 F0 00 3E 11 06       += 0398
1B10  1E CD 42 09 DD 21 D9 40 CD 48 0A C9 E5 CD 1F 02       += 0708
1B20  3E 00 32 68 40 32 67 40 CD 33 02 3E 01 32 66 40       += 040A
1B30  32 65 40 21 00 00 22 5F 40 22 61 40 22 63 40 22       += 0363
1B40  59 40 22 5B 40 22 5D 40 3E 01 32 58 40 E1 C9 50       += 0518
1B50  00 DC 00 33 00 73 74 61 72 74 65 6E 00 14 00 BE       += 04E2
1B60  00 22 00 41 64 72 3A 20 20 20 00 21 5D 1C CD DD       += 0417
1B70  06 21 44 00 11 BE 00 3E 22 06 1E CD 42 09 DD 21       += 03D4
1B80  D9 40 CD 48 0A C9 21 4F 1C CD DD 06 CD 6B 1C D8       += 0769
```

zu Abb. 8.2.11

```
------------- Seite  10  ------- Datei Grund 2.0 -----------
Rom                                                         Checksumme
1B90  CD E5 0A D8 CD 1C 1C 11 9C 1C D5 E9 AF 32 67 40      += 07A8
1BA0  32 68 40 CD 33 02 21 E4 1C CD DD 06 CD 50 01 FE      += 06C9
1BB0  46 C2 CF 1C 3A 48 40 B7 C2 C0 1C 3E 01 C3 C1 1C      += 06E9
1BC0  AF 32 48 40 32 49 40 32 67 40 CD 33 02 3E 46 FE      += 0581
1BD0  4D C2 AC 1C AF 32 48 40 32 46 40 3E 00 32 67 40      += 050F
1BE0  CD 33 02 C9 90 01 F5 00 11 00 46 3D 46 6C 69 70      += 0570
1BF0  20 4D 3D 4D 65 6E 75 65 00 21 0A 00 11 A0 00 3E      += 03BE
1C00  11 CD 72 07 2A 6E 40 7D E6 F0 6F 0E 08 C5 DD 21      += 06CA
1C10  80 40 CD 5E 09 06 10 CD BA 09 ED 5B 72 40 EB AF      += 072E
1C20  ED 52 7D B4 C2 2C 1D 3E 3E C3 2E 1D 3E 20 EB CD      += 071B
1C30  C5 09 7E CD 63 09 23 05 C2 17 1D E5 21 7A 40 CD      += 0630
1C40  DD 06 E1 11 10 00 AF ED 52 E5 2A 7C 40 11 0A 00      += 05B9
1C50  AF ED 52 22 7C 40 E1 DD 21 80 40 06 04 CD BA 09      += 0705
1C60  05 C2 5D 1D 06 10 CD BA 09 CD BA 09 7E 23 E6 7F      += 067D
1C70  FE 80 30 05 FE 20 D2 7B 1D 3E 20 CD C5 09 CD BA      += 07BB
1C80  09 05 C2 66 1D E5 21 7A 40 CD DD 06 2A 7C 40 11      += 05BA
1C90  0A 00 AF ED 52 22 7C 40 E1 C1 0D C2 0D 1D C9 50      += 068A
1CA0  00 DC 00 33 00 53 70 65 69 63 68 65 72 20 61 6E      += 0531
1CB0  73 65 68 65 6E 00 14 00 BE 00 22 00 2B 3D 77 65      += 044B
1CC0  69 74 65 72 20 2D 3D 72 75 65 63 6B 77 20 52 3D      += 057E
1CD0  41 64 72 20 4D 3D 4D 65 6E 75 65 00 0A 00 AE 00      += 0473
1CE0  21 00 2D 2D 20 30 20 31 20 32 20 33 20 34 20 35      += 026A
1CF0  20 36 20 37 20 38 20 39 20 41 20 42 20 43 20 44      += 02E8
1D00  20 45 20 46 00 CD 65 1E CD 6B 1C 22 72 40 22 6E      += 04D3
1D10  40 DA 64 1E E5 21 64 1E E3 CD F9 1C 21 B6 1D CD      += 07AA
1D20  DD 06 CD 50 01 E6 7F FE 4D C8 FE 2B C2 3C 1E 2A      += 07E8
1D30  6E 40 11 40 00 19 22 6E 40 C3 61 1E FE 2D C2 50      += 0567
1D40  1E 2A 6E 40 11 40 00 AF ED 52 22 6E 40 C3 61 1E      += 0547
1D50  FE 52 C2 61 1E CD 6B 1C DA 61 1E 22 72 40 22 6E      += 06A2
1D60  40 C3 19 1E C9 21 9F 1D CD DD 06 21 DC 1D CD DD      += 0754
1D70  06 C9 21 32 00 11 0A 00 3E 33 06 02 CD 42 09 3A      += 0308
1D80  D8 40 FE 01 C2 72 1E 3A D9 40 CD 53 01 FE 35 30      += 0740
1D90  05 FE 31 D2 9B 1E FE 57 C2 72 1E C9 55 AA 01 80      += 07AF
1DA0  AF 32 4A 40 32 46 40 32 48 40 21 9C 1E 11 C5 41      += 04CF
1DB0  06 04 E5 21 C9 1E E3 1A BE CA BE 1E 37 C9 23 13      += 068E
1DC0  05 C2 C6 1E AF C9 C3 B7 1E D2 F4 1E AF 32 C9 41      += 088A
1DD0  21 C9 41 22 C3 41 21 00 00 22 C1 41 21 9C 1E 11      += 0482
1DE0  C5 41 01 04 00 ED B0 21 08 40 11 09 40 01 17 00      += 0383
1DF0  36 00 ED B0 3E 07 CD 01 02 3E 00 CD 01 02 3E 02      += 0436
1E00  CD 01 02 CD 1F 02 21 1F 1F CD DD 06 06 64 CD 2B      += 052F
1E10  03 DA 0E 1F 05 C2 0E 1F CD 1F 02 CD DC 01 C9 32      += 0591
1E20  00 78 00 33 00 28 43 29 31 39 38 33 20 52 6F 6C      += 0361
1E30  66 2D 44 69 65 74 65 72 20 4B 6C 65 69 6E 00 28      += 052B
1E40  00 DC 00 44 00 52 44 4B 2D 47 72 75 6E 64 70 72      += 0510
1E50  6F 67 72 61 6D 6D 00 31 20 3D 20 61 65 6E 64 65      += 052E
1E60  72 6E 0A 32 20 3D 20 73 74 61 72 74 65 6E 0A 33      += 04D7
1E70  20 3D 20 61 6E 73 65 68 65 6E 0A 34 20 3D 20 53      += 046D
1E80  79 6D 62 6F 6C 65 0A 57 20 3D 20 77 65 69 74 65      += 0584
1E90  72 00 31 20 3D 20 4C 61 64 65 6E 20 43 41 53 0A      += 0405
```

zu Abb. 8.2.11

```
-------------- Seite  11  ------- Datei Grund 2.0 -----------
Rom                                                               Checksumme
1EA0  32 20 3D 20 53 70 65 69 63 68 65 72 6E 20 43 41             += 04F4
1EB0  53 0A 33 20 3D 20 50 72 75 65 66 65 6E 20 43 41             += 0486
1EC0  53 0A 57 20 3D 20 77 65 69 74 65 72 00 31 20 3D             += 044F
1ED0  20 49 4F 20 6C 65 73 65 6E 0A 32 20 3D 20 49 4F             += 0440
1EE0  20 73 65 74 7A 65 6E 0A 33 20 3D 20 45 69 6E 7A             += 0509
1EF0  65 6C 73 63 68 72 69 74 74 0A 57 20 3D 20 77 65             += 058C
1F00  69 74 65 72 00 31 20 3D 20 45 50 52 4F 4D 20 70             += 0475
1F10  72 6F 67 0A 32 20 3D 20 45 50 52 4F 4D 20 6C 65             += 0475
1F20  73 65 6E 00 21 32 00 11 B4 00 3E 33 CD 13 07 CD             += 0483
1F30  72 1E F5 CD 1F 02 F1 C9 31 C0 41 CD A0 1E CD 50             += 0807
1F40  0F E5 21 FE 20 E3 CD 1F 02 21 3F 1F CD DD 06 DD             += 0710
1F50  21 57 1F CD 24 20 FE 31 C2 61 20 CD 28 18 C3 FB             += 06E5
1F60  20 FE 32 C2 6C 20 CD 86 1C C3 FB 20 FE 33 C2 77             += 0855
1F70  20 CD 05 1E C3 FB 20 FE 34 C2 82 20 CD AE 16 C3             += 07D8
1F80  FB 20 FE 57 C2 FB 20 DD 21 92 1F CD 24 20 FE 31             += 083C
1F90  C2 99 20 CD CA 12 C3 FB 20 FE 32 C2 A4 20 CD 9E             += 0923
1FA0  14 C3 FB 20 FE 33 C2 AF 20 CD D0 12 C3 FB 20 FE             += 093F
1FB0  57 C2 FB 20 DD 21 CD 1F CD 24 20 FE 31 C2 C6 20             += 0806
1FC0  CD 89 15 C3 FB 20 FE 32 C2 D1 20 CD 3A 16 C3 FB             += 0907
1FD0  20 FE 33 C2 DC 20 CD 3F 1B C3 FB 20 FE 57 C2 FB             += 0926
1FE0  20 DD 21 05 20 CD 24 20 FE 31 C2 F3 20 CD FB 11             += 0731
1FF0  C3 FB 20 FE 32 C2 FB 20 CD B7 11 C3 46 20 76 00             += 081F
```

Abb. 8.2.11 Hexdump des Grundprogramms 2.0

8.3 Der Zeilenassembler und Disassembler (Debugger 2.1)

Bisher war die Programmentwicklung mit dem Grundprogramm doch sehr mühsam, denn man mußte Befehle im Maschinencode eingeben. Mit diesem Programm soll das anders werden. Es arbeitet allerdings nur zusammen mit dem Grundprogramm und daher kann man es nur auf einer ROA64 verwenden. Es beginnt ab Adresse 6000h bis 7FFFh und kommt daher in den Sockel 3 der ROA64. Das Grundprogramm bleibt auf Adresse 0.

Die Sockel 1 und 2 sind für weitere Programme reserviert.

Der Debugger wird gestartet, indem man vom Grundprogramm aus die Adresse 6000h (Menü „starten" und 6 0 0 0 CR eingeben) aufruft.

Er meldet sich mit einem kleinen Menü, wie in *Abb. 8.3.1* zu sehen ist.

```
Debugger V2.1 (C) 1984 Rolf-Dieter Klein

 1=Assembler
 2=Disassembler
 3=Starten
 4=Einzelschritt
 5=Protokoll an/aus
 6=Grundprogramm

Eingabe >>_
```

Abb. 8.3.1 So meldet sich der Debugger

Der Debugger bietet verschiedene Funktionen.

1. Assembler
   Damit kann man Zeile für Zeile Programme eingeben, ähnlich wie beim Grundprogramm, jedoch können Befehle auch in der Z80-Schreibweise direkt eingetippt werden. Die Übersetzung wird automatisch vorgenommen.
2. Disassembler
   Damit kann ein Maschinenprogramm in die Z80-Mnemonics rückübersetzt werden. Das ist für die Fehlersuche ganz praktisch.
3. Starten
   Start eines Programms, wie im Grundprogramm-Menü.
4. Einzelschritt
   Hier wird nur das Unterprogramm im Grundprogramm aufgerufen, um einen Test auf einfache Weise zu erlauben, ohne daß man in das Grundprogramm zurückkehren muß.
5. Protokoll an/aus.
   Wenn man einen Drucker besitzt, so kann man diesen über eine Centronics-Schnittstelle anschließen (siehe Kapitel IOE). Wenn man diese Funktion aufruft, so werden alle Aktionen des Assemblers oder Disassemblers auch auf den Drucker protokolliert. Ein erneuter Aufruf schaltet die Funktion wieder aus.
6. Grundprogramm
   Rückkehr ins Grundprogramm.

Assembler:

Abb. 8.3.2 zeigt eine Bildschirmkopie während gerade ein Programm mit dem Assembler eingegeben wurde.

Der Assembler meldet sich zunächst nur mit einem Cursor, der hier als Unterstreichungsstrich dargestellt wird. Man kann dann Befehle eingeben. Normalerweise beginnt der Assembler bei der Adresse 8800h, also dem ersten benutzbaren Speicherplatz.

Man kann dann eine Zeile eingeben. Dabei wird noch nicht kontrolliert, ob man Fehler macht. Fehlerhaft eingegebene Zeichen kann man löschen, indem man CTRL-H oder DEL eingibt. Erst wenn man die Taste CR drückt wird die Zeile analysiert.

Beginnen wir mit den einfachen Befehlen.

Wenn als erster Buchstabe ein „+“ steht, so wird die angezeigte Adresse um Eins erhöht. Im unteren Bildschirmteil wird übrigens immer eine rückübersetzte Version des dort liegenden Speicherinhalts angeboten.

Wenn man ein „–“ eingibt, so kann man eine Adresse rückwärts schreiten. Gibt man einen „.“ ein, so wird der Bildschirm gelöscht. Dies ist bei manchen Eingaben ganz praktisch, geschieht aber auch automatisch, wenn man am unteren Bildschirmrand ankommt.

Achtung, die Zahlenangaben unterscheiden sich vom Grundprogramm. Wenn man hier eine Zahl verwendet, so wird sie dezimal verstanden. Will man eine sedezimale Zahl eingeben, so muß

```
8800            start:
8800             anzahl:=5
8800 210500       ld hl,anzahl
8803 CD0F00       call schleife
8806 211400        ld hl,20
8809 CD0300        call schreite
880C 214800        ld hl,72
880F CD0600        call drehe
8812 CD1200       call endschleife
8815 C9           ret
8816            end_
```

Abb. 8.3.2 Eingabe eines Programms mit dem Zeilenassembler

```
8816                text:
8816 0A006400        defw 10,100
881A 33              defb 33h
881B 00              defb 0
881C 48616C6C        defb 'Hallo Test',0
     6F205465
     737400
8827                startetext:
8827 211688          ld hl,text
882A CD1E00          call write
882D C9              ret
882E                 end_
```

Abb. 8.3.3 Verschiedene Pseudobefehle

man den Buchstaben „h" anfügen und ggf. wenn die Zahl mit einem Buchstaben anfängt, wie z. B. FFh, auch eine 0 vorne anfügen, also 0FFh schreiben.

Wenn man ein Symbol definieren will, so schreibt man den Namen einfach hin, gefolgt von einem Doppelpunkt. Folgt dann nichts weiter, so wird die aktuelle Speicherzelle dem Namen zugewiesen. Dies entspricht dem NAME: = $ im Grundprogramm. Will man eine Zuweisung durchführen, so kann man auch wie im Grundprogramm verfahren, also NAME: = 123 oder wie im Bild ANZAHL: = 5. Wenn man ein Symbol zweimal definiert, so wird eine Fehlermeldung ausgegeben. Man kann dann z. B. das alte Symbol mit NAME: = % löschen.

Der Assembler kennt auch ein paar Pseudobefehle (siehe Kapitel 8.1):

DEFB
Damit werden Datenbytes abgelegt. Dabei kann man auch Texte abspeichern, ein Beispiel zeigt *Abb. 8.3.3*

DEFW
Ablage von Worten. Damit kann man z. B. Adressen speichern, dabei wird der niederwertige Teil zuerst abgelegt, so wie es der Z80 verlangt.

DEFS
Reservieren von Speicherbereichen.

Der Befehl DEFS 100 reserviert 100 Bytes, ohne jedoch Werte dorthin abzulegen.

ORG
Damit kann man eine neue Startadresse angeben. ORG 8800h legt die Startadresse auf den Wert 8800h.

OFF
Offset definieren. Will man ein Programm schreiben, das später einmal ab Adresse 0 laufen soll, so kann man dies tun, indem man schreibt:

```
ORG 0
OFF 8800h
```

Damit werden alle absoluten Adressen für den Bereich ab 0 ausgerechnet, der Maschinencode wird jedoch um 8800h Byte verschoben abgespeichert. Dann kann man ein EPROM programmieren und dieses anstelle des Grundprogramms auf Adresse 0 legen.

```
8800 210500    START: LD HL,ANZAHL
8803 CD0F00     CALL SCHLEIFE
8806 211400     LD HL,0014H
8809 CD0300     CALL SCHREITE
880C 214800     LD HL,0048H
880F CD0600     CALL DREHE
8812 CD1200     CALL ENDSCHLEIFE
8815 C9         RET

8816 00         NOP
```

Abb. 8.3.4 Ausgabe des Disassemblers

In der Testphase verwendet man den Offset-Befehl nicht, denn das Programm ist auf Adresse 8800h natürlich so nicht lauffähig.

END
Ende der Programmeingabe. Danach gelangt man wieder in das Menü des Debuggers. Wenn man danach wieder den Assembler aufruft, so wird die zuletzt vorhandene Startadresse genommen. Damit kann man Programme elegant Stück für Stück setzen.

Der Befehl EQU, wie er in anderen Assemblern vorkommt, wurde hier nicht verwendet. Dazu verwendet man hier die Zuweisung „:=".

Nun kann man nach erfolgter Eingabe das Programm mit Hilfe des Disassemblers noch mal kontrollieren. Der Disassembler fragt nach einer Start- und einer Endadresse. Achtung, wenn man eine sedezimale Eingabe machen will, so muß man auch hier ein „h" hinter die Zahl setzen. *Abb. 8.3.4* zeigt einen Beispielausdruck. Nach jedem RET-Befehl wird automatisch eine Leerzeile zur besseren Trennung eingefügt. Die Ausgabe kann man übrigens vorzeitig durch Eingabe von CTRL-C beenden. Nach jeder Seite wartet der Disassembler auf das Drücken einer Taste.

Zahlenrechnungen:
Da man häufig mit Zahlen rechnet, kennt der Assembler ein paar Operationen mehr als das Grundprogramm. Neben der Addition „+" und Subtraktion „−" kann man hier auch eine Oder-Verknüpfung mit „!" und eine Und-Verknüpfung mit „&" durchführen. Ferner gibt es das Einerkomplement durch den Operator „¬" (Achtung: Taste ß bei deutscher Tastatur). Rechnungen können auch mit Klammern verschachtelt werden.

Das Zeichen „$" steht auch hier für die aktuelle Speicheradresse. Mit einem Zeichen in Anführungszeichen kann man ebenfalls rechnen. Beispiel:

LD A,"A"+1

lädt den Wert des Zeichens „B" in das Register A.

8.3.1 Listing des Debuggers

Abb. 8.3.5 zeigt den Hexdump des Debuggers. Man kann das fertige EPROM aber auch von der Herstellerfirma der Bausätze beziehen (siehe Anhang).

```
-------------- Seite  1  ------- Datei Assem 2.1 -----------
Rom  Abs                                                          Checksumme
0000 6000  C3 C3 7D C3 24 00 C3 A9 60 C3 19 60 C3 D1 71 C3   += 08BA
0010 6010  D3 62 C3 32 73 C3 D0 61 C9 DB 49 0F 38 FB 79 D3   += 090C
0020 6020  48 AF D3 49 3E 01 D3 49 79 C9 F5 DB 70 E6 04 28   += 0802
0030 6030  FA F1 C9 CD 2A 60 D3 70 C9 3E 01 C3 33 60 3E 00   += 07EA
0040 6040  C3 33 60 CD 7F 60 CD 3E 60 C3 59 60 CD 7F 60 CD   += 0862
0050 6050  39 60 CD 59 60 CD 3E 60 C9 3E 03 CD 33 60 3E AC   += 06DE
0060 6060  CD 33 60 3E 02 CD 33 60 06 03 3E F8 CD 33 60 05   += 05A4
0070 6070  C2 6A 60 C9 CD 30 00 CD 2A 60 3E 00 D3 60 C9 E5   += 07C8
0080 6080  D5 3A 50 9F 4F 6F 06 00 60 29 09 29 29 E5 3A 51   += 0516
0090 6090  9F 4F 69 06 00 60 29 29 09 29 11 F6 00 EB AF ED   += 05CF
00A0 60A0  52 EB E1 CD 18 00 D1 E1 C9 3A 4F 9F B7 CA BB 60   += 0942
00B0 60B0  C5 79 FE 1A CA BA 60 CD 19 60 C1 C5 CD 4C 60 C1   += 0940
00C0 60C0  79 E6 7F FE 1A C2 D5 60 CD 74 60 AF 32 50 9F 32   += 0890
00D0 60D0  51 9F C3 4C 61 FE 0D C2 E1 60 AF 32 50 9F C3 4C   += 084D
00E0 60E0  61 FE 0A C2 04 61 3A 51 9F C6 01 32 51 9F FE 17   += 06B8
00F0 60F0  CA 01 61 DA 01 61 CD 74 60 3E 00 32 51 9F 32 50   += 05EB
0100 6100  9F C3 4C 61 FE 08 C2 1D 61 3A 50 9F D6 01 32 50   += 06D7
0110 6110  9F FE FF C2 1A 61 AF 32 50 9F C3 4C 61 FE 20 DA   += 0911
0120 6120  4C 61 F5 CD 7F 60 CD 39 60 3E 0A CD 33 60 CD 3E   += 0767
0130 6130  60 CD 7F 60 F1 CD 33 60 3A 50 9F 3C 32 50 9F FE   += 07E1
0140 6140  27 CA 4C 61 DA 4C 61 3E 27 32 50 9F CD 43 60 C9   += 06E4
0150 6150  CD 5A 62 CD 17 62 CD 55 62 CD D0 61 CD 3D 62 DD   += 089A
0160 6160  21 00 9F 2A 54 9F CD 5F 62 CD 17 62 DD 21 0E 9F   += 065C
0170 6170  16 00 CD 03 60 FE 0D CA B3 61 FE 08 20 18 7A B7   += 069E
0180 6180  28 F0 15 DD 2B 0E 08 CD 06 60 0E 20 CD 06 60 0E   += 04ED
0190 6190  08 CD 06 60 18 DC FE 7F 28 E4 FE 20 38 D4 DD 77   += 0836
01A0 61A0  00 7A FE 3D CA 72 61 DD 4E 00 CD 06 60 14 DD 23   += 06C4
01B0 61B0  C3 72 61 DD 36 00 00 C9 E5 CD 5F 62 DD 36 00 20   += 0718
01C0 61C0  DD 23 CD 2A 0D 7E CD 64 62 23 05 C2 C5 61 E1 C9   += 07CF
01D0 61D0  CD 3D 62 2A 54 9F DD 21 00 9F E5 CD B8 61 E1 DD   += 08AF
01E0 61E0  21 0E 9F CD 32 73 22 67 9F 3A 50 9F F5 3A 51 9F   += 06B0
01F0 61F0  F5 AF 32 50 9F 3E 18 32 51 9F 21 00 9F 06 27 C5   += 05EF
0200 6200  4E CD BB 60 23 C1 05 C2 FF 61 F1 32 51 9F F1 32   += 0877
0210 6210  50 9F C9 16 4B 18 02 16 0E 21 00 9F 4E 23 CD 06   += 045B
0220 6220  60 15 20 F8 C9 21 05 9F 22 4C 9F AF 32 4E 9F 21   += 0617
0230 6230  00 9F 11 01 9F 01 0D 00 36 20 ED B0 C9 21 05 9F   += 04DF
0240 6240  22 4C 9F AF 32 4E 9F 21 00 9F 11 01 9F 01 4A 00   += 0497
0250 6250  36 20 ED B0 C9 0E 0A CD 06 60 0E 0D C3 06 60 7C   += 05C7
0260 6260  CD 64 62 7D F5 0F 0F 0F 0F E6 0F CD 71 62 F1 E6   += 07AD
0270 6270  0F FE 0A 38 0A D6 0A C6 41 DD 77 00 DD 23 C9 C6   += 0723
0280 6280  30 DD 77 00 DD 23 C9 7D E5 CD 95 62 E1 7C CD 95   += 0932
0290 6290  62 C9 3A 58 9F F5 3A 4E 9F 3C 32 4E 9F FE 05 38   += 070E
02A0 62A0  12 D5 E5 CD 5A 62 CD 17 62 CD 55 62 CD 25 62 E1   += 0854
02B0 62B0  D1 18 E3 F1 E5 D5 2A 56 9F E5 ED 5B 52 9F 19 77   += 0944
02C0 62C0  E1 23 22 56 9F D1 E1 DD 2A 4C 9F CD 64 62 DD 22   += 0851
02D0 62D0  4C 9F C9 CD 02 70 FD 7E 00 FE 00 CA 75 63 FE 3B   += 0847
02E0 62E0  CA 75 63 01 20 20 11 20 20 FD 7E 00 CD 65 63 DA   += 061E
02F0 62F0  78 63 4F FD 23 FD 7E 00 CD 65 63 DA 22 63 47 FD   += 07FD
0300 6300  23 FD 7E 00 CD 65 63 DA 22 63 5F FD 23 FD 7E 00   += 078C
```

zu Abb. 8.3.5

```
-------------- Seite  2  ------- Datei Assem 2.1 -----------
Rom  Abs                                                       Checksumme
0310 6310  CD 65 63 DA 22 63 57 FD 23 FD 7E 00 CD 65 63 D2   += 084D
0320 6320  78 63 DD 21 7A 63 DD 7E 00 B7 CA 78 63 B9 20 1F   += 0765
0330 6330  DD 7E 01 B8 20 19 DD 7E 02 BB 20 13 DD 7E 03 BA   += 06B0
0340 6340  20 0D DD 7E 04 32 58 9F DD 6E 05 DD 66 06 E9 D9   += 0710
0350 6350  01 07 00 DD 09 D9 18 CE CD 65 63 D0 FE 30 D8 FE   += 0816
0360 6360  3A 38 12 37 C9 FE 41 D8 FE 5B 38 09 FE 61 D8 FE   += 086A
0370 6370  7B 30 05 D6 20 37 3F C9 37 C9 4A 50 20 20 C2 8B   += 060C
0380 6380  6F 43 41 4C 4C C4 87 6F 44 4A 4E 5A 10 32 6F 52   += 057E
0390 6390  45 54 20 C0 7D 6F 4A 52 20 20 18 34 6F 43 43 46   += 04C8
03A0 63A0  20 3F FC 6F 53 43 46 20 37 FC 6F 43 50 4C 20 2F   += 0596
03B0 63B0  FC 6F 44 41 41 20 27 FC 6F 44 49 20 20 F3 FC 6F   += 070E
03C0 63C0  45 49 20 20 FB FC 6F 45 58 58 20 D9 FC 6F 48 41   += 0716
03D0 63D0  4C 54 76 FC 6F 4E 4F 50 20 00 FC 6F 52 4C 41 20   += 05F8
03E0 63E0  17 FC 6F 52 4C 43 41 07 FC 6F 52 52 41 20 1F FC   += 0636
03F0 63F0  6F 52 52 43 41 0F FC 6F 43 50 44 20 A9 1B 6F 43   += 057E
0400 6400  50 44 52 B9 1B 6F 43 50 49 20 A1 1B 6F 43 50 49   += 052C
0410 6410  52 B1 1B 6F 49 4E 44 20 AA 1B 6F 49 4E 44 52 BA   += 05A3
0420 6420  1B 6F 49 4E 49 20 A2 1B 6F 49 4E 49 52 B2 1B 6F   += 0524
0430 6430  4C 44 44 20 A8 1B 6F 4C 44 44 52 B8 1B 6F 4C 44   += 051E
0440 6440  49 20 A0 1B 6F 4C 44 49 52 B0 1B 6F 4F 54 44 52   += 0531
0450 6450  BB 1B 6F 4F 54 49 52 B3 1B 6F 4F 55 54 44 AB 1B   += 05C2
0460 6460  6F 4F 55 54 49 A3 1B 6F 52 45 54 49 4D 1B 6F 52   += 053A
0470 6470  45 54 4E 45 1B 6F 52 4C 44 20 6F 1B 6F 52 52 44   += 0499
0480 6480  20 67 1B 6F 4E 45 47 20 44 1B 6F 52 53 54 20 C7   += 04B9
0490 6490  F5 6E 49 4D 20 20 46 C3 6E 41 4E 44 20 A0 57 6E   += 0608
04A0 64A0  4F 52 20 20 B0 57 6E 58 4F 52 20 A8 57 6E 43 50   += 056F
04B0 64B0  20 20 B8 57 6E 53 55 42 20 90 57 6E 41 44 44 20   += 0505
04C0 64C0  80 9C 6D 41 44 43 20 88 9C 6D 53 42 43 20 98 9C   += 062E
04D0 64D0  6D 45 58 20 20 E3 10 6D 49 4E 20 20 DB 5D 6C 4F   += 0574
04E0 64E0  55 54 20 D3 AA 6C 50 55 53 48 C5 1D 6C 50 4F 50   += 062F
04F0 64F0  20 C1 1D 6C 42 49 54 20 40 86 6B 53 45 54 20 C0   += 0566
0500 6500  86 6B 52 45 53 20 80 86 6B 52 4C 43 20 00 AB 6B   += 0583
0510 6510  52 52 43 20 08 AB 6B 52 4C 20 20 10 AB 6B 52 52   += 04CD
0520 6520  20 20 18 AB 6B 53 4C 41 20 20 AB 6B 53 52 41 20   += 04AA
0530 6530  28 AB 6B 53 52 4C 20 38 AB 6B 49 4E 43 20 04 D1   += 056C
0540 6540  6A 44 45 43 20 05 D1 6A 4C 44 20 20 40 64 67 4F   += 04C0
0550 6550  52 47 20 00 E7 66 44 45 46 42 00 F1 66 44 45 46   += 053D
0560 6560  57 00 46 67 44 45 46 53 00 56 67 4F 46 46 20 00   += 03DE
0570 6570  DD 66 45 4E 44 20 00 F6 6F 00 00 00 00 00 CD 02   += 046E
0580 6580  70 FD 22 5B 9F FD 7E 00 CD 65 63 D8 4F 06 20 FD   += 07E3
0590 6590  23 FD 7E 00 CD 65 63 38 0C 47 FD 23 FD 7E 00 CD   += 0726
05A0 65A0  65 63 D2 BE 65 DD 7E 00 B7 CA BE 65 B9 20 16 DD   += 0888
05B0 65B0  7E 01 B8 20 10 DD 7E 02 32 59 9F C3 75 63 FD 2A   += 06B0
05C0 65C0  5B 9F C3 78 63 D5 11 03 00 DD 19 D1 18 D7 4E 5A   += 06DF
05D0 65D0  00 5A 20 08 4E 43 10 43 20 18 50 4F 20 50 45 28   += 031A
05E0 65E0  50 20 30 4D 20 38 00 00 00 CD 02 70 FD 7E 00 FE   += 04FD
05F0 65F0  2B 18 1C CD 02 70 FD 7E 00 FE 28 18 12 CD 02 70   += 05A8
0600 6600  FD 7E 00 FE 29 18 08 CD 02 70 FD 7E 00 FE 2C C2   += 0768
0610 6610  78 63 FD 23 C3 75 63 42 20 00 43 20 01 44 20 02   += 04C2
```

zu Abb. 8.3.5

```
-------------- Seite  3  ------- Datei Assem 2.1 -----------
Rom  Abs                                                          Checksumme
0620 6620  45 20 03 48 20 04 4C 20 05 41 20 07 00 00 00 48   += 01F5
0630 6630  4C 00 49 58 01 49 59 02 00 00 00 42 43 00 44 45   += 02A0
0640 6640  10 48 4C 20 53 50 30 00 00 00 42 43 00 44 45 10   += 02B5
0650 6650  49 58 20 53 50 30 00 00 00 42 43 00 44 45 10 49   += 02FB
0660 6660  59 20 53 50 30 00 00 00 49 58 02 49 59 03 00 00   += 0294
0670 6670  00 42 43 00 44 45 10 48 4C 20 41 46 30 53 50 01   += 032D
0680 6680  49 58 02 49 59 03 00 00 00 49 20 00 52 20 01 00   += 0224
0690 6690  00 00 DD 21 71 66 C3 7E 65 DD 21 17 66 C3 7E 65   += 069C
06A0 66A0  DD 21 2F 66 C3 7E 65 DD 21 3B 66 C3 7E 65 DD 21   += 077C
06B0 66B0  4A 66 C3 7E 65 DD 21 59 66 C3 7E 65 DD 21 68 66   += 0785
06C0 66C0  C3 7E 65 DD 21 89 66 C3 7E 65 CD 02 70 FD 7E 00   += 07F3
06D0 66D0  CD 65 63 FE 41 C2 78 63 FD 23 C3 75 63 CD 0C 70   += 0875
06E0 66E0  D8 22 52 9F C3 75 63 CD 0C 70 D8 22 56 9F C3 75   += 07F6
06F0 66F0  63 CD 02 70 FD 7E 00 FE 27 C2 16 67 FD 23 FD 7E   += 081C
0700 6700  00 FE 27 CA 11 67 CD 95 62 FD 23 FD 7E 00 C3 01   += 078A
0710 6710  67 FD 23 C3 3D 67 FE 22 C2 35 67 FD 23 FD 7E 00   += 0807
0720 6720  FE 22 CA 30 67 CD 95 62 FD 23 FD 7E 00 C3 20 67   += 082A
0730 6730  FD 23 C3 3D 67 CD 0C 70 D8 7D CD 95 62 CD 07 66   += 0823
0740 6740  D2 F1 66 C3 75 63 CD 0C 70 D8 CD 87 62 CD 07 66   += 08D5
0750 6750  D2 46 67 C3 75 63 CD 0C 70 ED 4B 56 9F 09 22 56   += 0711
0760 6760  9F C3 75 63 CD F3 65 DA 95 68 CD A7 66 DA C3 67   += 0A14
0770 6770  5F CD FD 65 D8 CD 07 66 D8 7B FE 30 CA 78 63 B7   += 097D
0780 6780  C2 8F 67 CD CA 66 D8 3E 02 CD 95 62 C3 C0 67 FE   += 0979
0790 6790  10 C2 A0 67 CD CA 66 D8 3E 12 CD 95 62 C3 C0 67   += 08AC
07A0 67A0  FE 20 C2 C0 67 CD 99 66 DA B3 67 F6 70 CD 95 62   += 09F1
07B0 67B0  C3 C0 67 3E 36 CD 95 62 CD 0C 70 D8 7D CD 95 62   += 0884
07C0 67C0  C3 92 68 CD BC 66 DA 01 68 FE 02 C2 D3 67 3E DD   += 0906
07D0 67D0  C3 D5 67 3E FD CD 95 62 CD E9 65 D8 CD 0C 70 CD   += 0A07
07E0 67E0  6C 6F D8 22 5D 9F CD FD 65 D8 CD 07 66 D8 CD 99   += 0950
07F0 67F0  66 D8 F6 70 CD 95 62 2A 5D 9F 7D CD 95 62 C3 92   += 0924
0800 6800  68 CD 0C 70 D8 22 5D 9F CD FD 65 D8 CD 07 66 D8   += 08C0
0810 6810  CD CA 66 DA 1E 68 3E 32 CD 95 62 C3 8C 68 CD 92   += 08A7
0820 6820  66 DA 78 63 FE 30 CA 78 63 B7 C2 3A 68 3E ED CD   += 0901
0830 6830  95 62 3E 43 CD 95 62 C3 8C 68 FE 10 C2 4C 68 3E   += 07B5
0840 6840  ED CD 95 62 3E 53 CD 95 62 C3 8C 68 FE 20 C2 59   += 08F6
0850 6850  68 3E 22 CD 95 62 C3 8C 68 FE 01 C2 6B 68 3E ED   += 0802
0860 6860  CD 95 62 3E 73 CD 95 62 C3 8C 68 FE 02 C2 7D 68   += 0897
0870 6870  3E DD CD 95 62 3E 22 CD 95 62 C3 8C 68 FE 03 C2   += 087D
0880 6880  8C 68 3E FD CD 95 62 3E 22 CD 95 62 2A 5D 9F CD   += 080A
0890 6890  87 62 C3 75 63 CD 99 66 DA BD 69 5F CD 07 66 DA   += 08C3
08A0 68A0  78 63 CD 99 66 DA BF 68 E6 07 F6 40 57 7B 07 07   += 07AB
08B0 68B0  07 E6 38 B2 FE 76 CA 78 63 CD 95 62 C3 BA 69 CD   += 0967
08C0 68C0  C3 66 DA F4 68 B7 C2 DC 68 7B FE 07 C2 78 63 3E   += 0977
08D0 68D0  ED CD 95 62 3E 57 CD 95 62 C3 F1 68 FE 01 C2 F1   += 09D8
08E0 68E0  68 7B FE 07 C2 78 63 3E ED CD 95 62 3E 5F CD 95   += 0873
08F0 68F0  62 C3 BA 69 CD F3 65 DA A5 69 CD 92 66 D2 15 69   += 096A
0900 6900  7B FE 07 C2 78 63 3E 3A CD 95 62 CD 0C 70 D8 CD   += 0847
0910 6910  87 62 C3 9C 69 FE 20 C2 28 69 7B 07 07 07 E6 38   += 06D0
0920 6920  F6 46 CD 95 62 C3 9C 69 B7 C2 3A 69 7B FE 07 C2   += 0926
```

zu Abb. 8.3.5

```
-------------- Seite  4  ------- Datei Assem 2.1 -----------
Rom  Abs                                                          Checksumme
0930 6930  78 63 3E 0A CD 95 62 C3 9C 69 FE 10 C2 4D 69 7B  += 07B0
0940 6940  FE 07 C2 78 63 3E 1A CD 95 62 C3 9C 69 FE 02 C2  += 0848
0950 6950  73 69 3E DD CD 95 62 7B 07 07 07 E6 38 F6 46 CD  += 0772
0960 6960  95 62 CD E9 65 DA 78 63 CD 0C 70 D8 7D CD 95 62  += 0929
0970 6970  C3 9C 69 FE 03 C2 99 69 3E FD CD 95 62 7B 07 07  += 0815
0980 6980  07 E6 38 F6 46 CD 95 62 CD E9 65 DA 78 63 CD 0C  += 08CE
0990 6990  70 D8 7D CD 95 62 C3 9C 69 C3 78 63 CD FD 65 DA  += 09F8
09A0 69A0  78 63 C3 BA 69 7B 07 07 07 E6 38 F6 06 CD 95 62  += 072F
09B0 69B0  CD 0C 70 DA 78 63 7D CD 95 62 C3 CE 6A CD C3 66  += 0930
09C0 69C0  DA F6 69 B7 C2 DC 69 CD 07 66 D8 CD CA 66 D8 3E  += 0A1C
09D0 69D0  ED CD 95 62 3E 47 CD 95 62 C3 F3 69 FE 01 C2 F3  += 09CD
09E0 69E0  69 CD 07 66 D8 CD CA 66 D8 3E ED CD 95 62 3E 4F  += 08CC
09F0 69F0  CD 95 62 C3 CE 6A CD A7 66 DA 90 6A 5F CD 07 66  += 0906
0A00 6A00  D8 CD F3 65 DA 55 6A 7B B7 C2 19 6A 3E ED CD 95  += 099A
0A10 6A10  62 3E 4B CD 95 62 C3 47 6A FE 10 C2 2B 6A 3E ED  += 07B3
0A20 6A20  CD 95 62 3E 5B CD 95 62 C3 47 6A FE 20 C2 38 6A  += 0817
0A30 6A30  3E 2A CD 95 62 C3 47 6A FE 30 C2 47 6A 3E ED CD  += 0839
0A40 6A40  95 62 3E 7B CD 95 62 CD 0C 70 D8 CD 87 62 CD FD  += 0915
0A50 6A50  65 D8 C3 8D 6A CD A0 66 D2 6B 6A 7B F6 01 CD 95  += 0945
0A60 6A60  62 CD 0C 70 D8 CD 87 62 C3 8D 6A FE 01 C2 78 6A  += 0896
0A70 6A70  3E DD CD 95 62 C3 82 6A FE 02 C2 82 6A 3E FD CD  += 0944
0A80 6A80  95 62 3E F9 CD 95 62 7B FE 30 C2 78 63 C3 CE 6A  += 0933
0A90 6A90  CD BC 66 D8 5F CD 07 66 D8 7B FE 02 C2 A4 6A 3E  += 08C1
0AA0 6AA0  DD C3 A6 6A 3E FD CD 95 62 CD F3 65 DA C2 6A 3E  += 0A18
0AB0 6AB0  2A CD 95 62 CD 0C 70 D8 CD 87 62 CD FD 65 D8 C3  += 098F
0AC0 6AC0  CE 6A 3E 21 CD 95 62 CD 0C 70 D8 CD 87 62 C3 75  += 086A
0AD0 6AD0  63 CD 99 66 DA E7 6A 07 07 07 E6 38 47 3A 58 9F  += 0705
0AE0 6AE0  B0 CD 95 62 C3 75 63 CD F3 65 DA 34 6B CD 92 66  += 0972
0AF0 6AF0  DA 78 63 FE 20 C2 03 6B 3A 58 9F F6 30 CD 95 62  += 081E
0B00 6B00  C3 30 6B FE 02 C2 0D 6B 3E DD C3 0F 6B 3E FD CD  += 07F8
0B10 6B10  95 62 3A 58 9F F6 30 CD 95 62 CD E9 65 DA 78 63  += 08E2
0B20 6B20  CD 0C 70 DA 78 63 CD 6C 6F DA 78 63 7D CD 95 62  += 089C
0B30 6B30  CD FD 65 C9 CD A7 66 D2 6F 6B CD 92 66 DA 78 63  += 09F8
0B40 6B40  FE 02 C2 4A 6B 3E DD C3 57 6B FE 03 C2 54 6B 3E  += 07D7
0B50 6B50  FD C3 57 6B C3 78 63 CD 95 62 3A 58 9F FE 04 C2  += 08D9
0B60 6B60  67 6B 3E 23 C3 69 6B 3E 2B CD 95 62 C3 83 6B 47  += 06EF
0B70 6B70  3A 58 9F FE 04 C2 7D 6B 3E 03 C3 7F 6B 3E 0B B0  += 06C4
0B80 6B80  CD 95 62 C3 75 63 CD 02 70 FD 7E 00 FE 30 DA 78  += 0899
0B90 6B90  63 FE 38 D2 78 63 FD 23 07 07 07 E6 38 47 3A 58  += 0672
0BA0 6BA0  9F B0 32 58 9F CD 07 66 DA 78 63 CD F3 65 D2 C4  += 0922
0BB0 6BB0  6B 3E CB CD 95 62 CD 99 66 47 3A 58 9F B0 CD 95  += 088E
0BC0 6BC0  62 C3 1A 6C CD 92 66 DA 78 63 FE 20 CA 07 6C FE  += 087E
0BD0 6BD0  02 C2 D9 6B 3E DD C3 E6 6B FE 03 C2 E3 6B 3E FD  += 0983
0BE0 6BE0  C3 E6 6B DA 78 63 CD 95 62 3E CB CD 95 62 CD E9  += 0A10
0BF0 6BF0  65 DA 78 63 CD 0C 70 DA 78 63 CD 6C 6F DA 78 63  += 0875
0C00 6C00  7D CD 95 62 C3 0C 6C 3E CB CD 95 62 3A 58 9F F6  += 0870
0C10 6C10  06 CD 95 62 CD FD 65 DA 78 63 C3 75 63 CD 92 66  += 090E
0C20 6C20  DA 78 63 FE 01 CA 78 63 FE 02 C2 3D 6C 3E DD CD  += 08AC
0C30 6C30  95 62 3A 58 9F F6 20 CD 95 62 C3 5A 6C FE 03 C2  += 084E
```

zu Abb. 8.3.5

```
-------------- Seite  5  ------- Datei Assem 2.1 -----------
Rom  Abs                                                        Checksumme
0C40 6C40  52 6C 3E FD CD 95 62 3A 58 9F F6 20 CD 95 62 C3    += 088B
0C50 6C50  5A 6C 47 3A 58 9F B0 CD 95 62 C3 75 63 CD 99 66    += 0819
0C60 6C60  DA 78 63 5F CD 07 66 DA 78 63 CD F3 65 DA 78 63    += 08DD
0C70 6C70  CD 02 70 FD 7E 00 CD 65 63 FE 43 C2 93 6C FD 23    += 0871
0C80 6C80  3E ED CD 95 62 7B 07 07 07 E6 38 F6 40 CD 95 62    += 0797
0C90 6C90  C3 A6 6C 7B FE 07 C2 78 63 CD 92 62 CD 0C 70 DA    += 08D6
0CA0 6CA0  78 63 7D CD 95 62 CD FD 65 C9 CD F3 65 DA 78 63    += 09EE
0CB0 6CB0  CD 02 70 FD 7E 00 CD 65 63 FE 43 C2 E4 6C 3E ED    += 08CD
0CC0 6CC0  CD 95 62 FD 23 CD FD 65 DA 78 63 CD 07 66 DA 78    += 0954
0CD0 6CD0  63 CD 99 66 DA 78 63 07 07 07 E6 38 F6 41 CD 95    += 07B0
0CE0 6CE0  62 C3 0D 6D CD 92 62 CD 0C 70 DA 78 63 7D CD 95    += 083D
0CF0 6CF0  62 CD FD 65 DA 78 63 CD 07 66 DA 78 63 CD 02 70    += 0874
0D00 6D00  FD 7E 00 CD 65 63 FE 41 C2 78 63 FD 23 C3 75 63    += 08A7
0D10 6D10  CD F3 65 D2 62 6D CD 92 66 DA 78 63 FE 30 C2 41    += 0971
0D20 6D20  6D CD 07 66 CD 92 66 DA 78 63 FE 30 C2 78 63 FD    += 08E9
0D30 6D30  7E 00 FE 27 C2 78 63 FD 23 3E 08 CD 95 62 C3 5F    += 078C
0D40 6D40  6D FE 10 C2 5C 6D CD 07 66 CD 92 66 DA 78 63 FE    += 08B8
0D50 6D50  20 C2 78 63 3E EB CD 95 62 C3 5F 6D C3 78 63 C3    += 089A
0D60 6D60  75 63 CD 92 66 DA 78 63 FE 01 C2 78 63 CD FD 65    += 091D
0D70 6D70  DA 78 63 CD 07 66 DA 78 63 CD A0 66 DA 78 63 FE    += 092A
0D80 6D80  01 C2 8C 6D 3E DD CD 95 62 C3 96 6D FE 02 C2 96    += 08B9
0D90 6D90  6D 3E FD CD 95 62 CD 92 62 C3 75 63 CD A0 66 DA    += 0975
0DA0 6DA0  41 6E B7 C2 F5 6D CD 07 66 DA 78 63 CD A7 66 DA    += 092D
0DB0 6DB0  78 63 3A 58 9F FE 80 C2 C5 6D 3A 59 9F F6 09 CD    += 087C
0DC0 6DC0  95 62 C3 F2 6D FE 88 C2 DA 6D 3E ED CD 95 62 3A    += 09D1
0DD0 6DD0  59 9F F6 4A CD 95 62 C3 F2 6D FE 98 C2 EF 6D 3E    += 0A10
0DE0 6DE0  ED CD 95 62 3A 59 9F F6 42 CD 95 62 C3 F2 6D C3    += 09C4
0DF0 6DF0  78 63 C3 3E 6E FE 01 C2 1B 6E 3A 58 9F FE 80 C2    += 0805
0E00 6E00  78 63 CD 07 66 DA 78 63 CD AE 66 3E DD CD 95 62    += 088A
0E10 6E10  3A 59 9F F6 09 CD 95 62 C3 3E 6E FE 02 C2 3E 6E    += 07D2
0E20 6E20  3A 58 9F FE 80 C2 78 63 CD 07 66 DA 78 63 CD B5    += 08BD
0E30 6E30  66 3E FD CD 95 62 3A 59 9F F6 09 CD 95 62 C3 75    += 0892
0E40 6E40  63 CD 02 70 FD 7E 00 CD 65 63 FE 41 C2 78 63 FD    += 088B
0E50 6E50  23 CD 07 66 DA 78 63 CD F3 65 38 42 CD A0 66 DA    += 085E
0E60 6E60  78 63 B7 C2 71 6E 3A 58 9F F6 06 CD 95 62 C3 95    += 087C
0E70 6E70  6E FE 01 C2 7B 6E 3E DD C3 7D 6E 3E FD CD 95 62    += 08E0
0E80 6E80  3A 58 9F F6 06 CD 95 62 CD E9 65 DA 78 63 CD 0C    += 089A
0E90 6E90  70 7D CD 95 62 CD FD 65 DA 78 63 C3 75 63 CD 99    += 0996
0EA0 6EA0  66 38 0B 47 3A 58 9F B0 CD 95 62 C3 75 63 3A 58    += 06C2
0EB0 6EB0  9F F6 46 CD 95 62 CD 0C 70 DA 78 63 7D CD 95 62    += 08DE
0EC0 6EC0  C3 75 63 3E ED CD 95 62 CD 02 70 FD 7E 00 FD 23    += 0864
0ED0 6ED0  FE 30 C2 DA 6E 3E 46 C3 F1 6E FE 31 C2 E4 6E 3E    += 095F
0EE0 6EE0  56 C3 F1 6E FE 32 C2 EE 6E 3E 5E C3 F1 6E C3 78    += 09BF
0EF0 6EF0  63 CD 95 62 C9 CD 0C 70 D8 7C B7 C2 78 63 7D FE    += 095C
0F00 6F00  08 D2 0D 6F 07 07 07 F6 C7 CD 95 62 C9 E6 C7 C2    += 0824
0F10 6F10  78 63 7D F6 C7 CD 95 62 C3 75 63 3E ED CD 95 62    += 0963
0F20 6F20  C3 FC 6F 43 20 38 4E 43 30 5A 20 28 4E 5A 20 00    += 04F4
0F30 6F30  00 00 18 1B CD 02 70 FD E5 DD 21 23 6F CD 7E 65    += 0694
0F40 6F40  DA 4D 6F 32 58 9F CD 07 66 E1 D8 18 02 FD E1 CD    += 0877
```

zu Abb. 8.3.5

```
------------- Seite  6 ------- Datei Assem 2.1 -----------
Rom  Abs                                                         Checksumme
0F50 6F50  92 62 CD 0C 70 D8 ED 5B 54 9F 13 13 AF ED 52 CD   += 0831
0F60 6F60  6C 6F DA 78 63 7D CD 95 62 C3 75 63 7D E6 80 B4   += 0903
0F70 6F70  28 08 7D F6 7F A4 3C C2 78 63 C3 75 63 0E C9 06   += 0717
0F80 6F80  01 CD C7 6F C3 75 63 0E CD 18 2C CD F3 65 DA B5   += 0872
0F90 6F90  6F CD A0 66 D8 FE 01 C2 A2 6F 3E DD CD 95 62 C3   += 098E
0FA0 6FA0  AC 6F FE 02 C2 AC 6F 3E FD CD 95 62 3E E9 CD 95   += 0980
0FB0 6FB0  62 CD FD 65 C9 0E C3 06 00 CD C7 6F D8 CD 0C 70   += 0855
0FC0 6FC0  D8 CD 87 62 C3 75 63 CD 02 70 FD E5 DD 21 CE 65   += 097B
0FD0 6FD0  C5 CD 7E 65 C1 38 15 4F 3A 58 9F B1 32 58 9F E1   += 07BE
0FE0 6FE0  78 B7 20 0E CD 07 66 DA 78 63 18 06 79 32 58 9F   += 060C
0FF0 6FF0  FD E1 CD 92 62 C9 3E 01 32 5A 9F C9 CD 92 62 C3   += 091F
1000 7000  75 63 FD 7E 00 FE 20 C0 FD 23 18 F6 CD 02 70 CD   += 086B
1010 7010  58 70 D8 CD 02 70 FD 7E 00 FE 2B C2 2A 70 FD 23   += 07FF
1020 7020  E5 CD 58 70 D1 D8 19 C3 56 70 FE 2D C2 3E 70 FD   += 095D
1030 7030  23 E5 CD 58 70 D1 D8 EB AF ED 52 C3 56 70 FE 21   += 09C7
1040 7040  C2 53 70 FD 23 E5 CD 58 70 D1 7B B5 6F 7A B4 67   += 0924
1050 7050  C3 56 70 C3 75 63 18 BB CD 92 70 D8 CD 02 70 FD   += 08DA
1060 7060  7E 00 FE 26 C2 78 70 FD 23 E5 CD 92 70 D1 D8 7B   += 0944
1070 7070  A5 6F 7A A4 67 C3 7B 70 C3 75 63 18 DF FE 30 D8   += 08DF
1080 7080  FE 3A 3F C9 CD 65 63 CD 7D 70 D0 FE 41 D8 FE 47   += 09BB
1090 7090  3F C9 CD 02 70 FD 7E 00 FE 24 20 08 FD 23 2A 54   += 06AA
10A0 70A0  9F C3 75 63 FE 28 20 15 FD 23 CD 0C 70 CD 02 70   += 073D
10B0 70B0  FD 7E 00 FE 29 C2 78 63 FD 23 C3 75 63 FE 7E 20   += 0896
10C0 70C0  0F FD 23 CD 92 70 D8 7D 2F 6F 7C 2F 67 C3 75 63   += 079E
10D0 70D0  FE 2D 20 10 FD 23 CD 92 70 D8 EB 21 00 00 AF ED   += 07CA
10E0 70E0  52 C3 75 63 FE 27 20 25 FD 23 FD 7E 00 6F 26 00   += 0687
10F0 70F0  FD 23 FD 7E 00 FD 23 FE 27 CA 75 63 65 6F FD 23   += 0876
1100 7100  FD 7E 00 FD 23 FE 27 C2 78 63 C3 75 63 CD 02 70   += 0837
1110 7110  FD E5 FD 7E 00 CD 7D 70 DD E1 DA 91 71 FD 23 FD   += 0ACE
1120 7120  7E 00 CD 84 70 D2 1D 71 FD 7E 00 DD E5 FD E1 FE   += 09B8
1130 7130  68 28 20 FE 48 28 1C 21 00 00 FD 7E 00 CD 7D 70   += 0590
1140 7140  DA 75 63 FD 23 E6 0F 4F 06 00 E5 D1 29 29 19 29   += 0666
1150 7150  09 18 E7 FD 7E 00 CD 7D 70 D8 21 00 00 FD 7E 00   += 06B1
1160 7160  CD 84 70 DA 80 71 FD 23 FE 41 38 08 FE 47 30 04   += 07A4
1170 7170  D6 41 C6 0A E6 0F 4F 06 00 29 29 29 29 09 18 DD   += 04D3
1180 7180  FD 23 FE 48 CA 75 63 FE 68 CA 75 63 FD 2B C3 78   += 0973
1190 7190  63 FD E5 DD E1 FD 21 C9 81 CD D0 0A DD E5 FD E1   += 0BB2
11A0 71A0  D0 FD E5 DD E1 FD 21 45 0A CD D0 0A DD E5 FD E1   += 0B24
11B0 71B0  C9 46 65 68 6C 65 72 3A 20 4E 61 6D 65 20 73 63   += 05F0
11C0 71C0  68 6F 6E 20 76 65 72 77 65 6E 64 65 74 2E 0D 0A   += 057E
11D0 71D0  00 AF 32 5A 9F CD 3D 62 CD 50 61 FD 21 0E 9F CD   += 075C
11E0 71E0  02 70 FD 7E 00 B7 C2 F7 71 2A 67 9F 22 54 9F 22   += 0735
11F0 71F0  56 9F 18 E4 C3 2B 72 FE 2B C2 0B 72 2A 54 9F 23   += 06F9
1200 7200  22 54 9F 22 56 9F 18 D0 C3 2B 72 FE 2D C2 1F 72   += 06F2
1210 7210  2A 54 9F 2B 22 54 9F 22 56 9F 18 BC C3 2B 72 FE   += 06A6
1220 7220  2E C2 2B 72 0E 1A CD 06 60 18 AD CD 65 63 DA C4   += 06E0
1230 7230  72 DA 3F 72 FD 23 FD 7E 00 CD 58 63 C3 31 72 FE   += 0884
1240 7240  3A C2 C4 72 FD 23 FD 7E 00 FE 3D 20 46 FD 23 CD   += 085B
1250 7250  02 70 FD 7E 00 FE 25 20 17 FD 23 FD E5 FD 21 0E   += 0775
```

zu Abb. 8.3.5

```
-------------- Seite  7  ------- Datei Assem 2.1 -----------
Rom  Abs                                                       Checksumme
1260 7260  9F CD 02 70 FD E5 DD E1 CD 70 0B FD E1 C3 CB 72  += 0AA4
1270 7270  CD 0C 70 DA CB 72 FD E5 E5 FD 21 0E 9F CD 02 70  += 0931
1280 7280  FD E5 DD E1 DD E5 CD 70 0B DD E1 E1 CD EE 0B FD  += 0C0C
1290 7290  E1 18 38 FD E5 FD 21 0E 9F CD 02 70 FD E5 DD E1  += 09BD
12A0 72A0  2A 54 9F CD EE 0B D2 C0 72 CD 5A 62 CD 17 62 CD  += 0883
12B0 72B0  55 62 CD 25 62 21 B1 71 CD F4 7C FD E1 C3 D8 71  += 0975
12C0 72C0  FD E1 18 04 FD 21 0E 9F CD D3 62 DA E7 72 CD 02  += 08C9
12D0 72D0  70 FD 7E 00 FE 3B CA E6 72 FE 00 CA E2 72 37 C3  += 095C
12E0 72E0  E3 72 AF C3 E7 72 AF DA F3 72 2A 56 9F 22 54 9F  += 0942
12F0 72F0  C3 27 73 2A 54 9F 22 56 9F FD E5 CD 5A 62 CD 17  += 07E0
1300 7300  62 CD 55 62 CD 3D 62 D1 21 00 9F E5 21 1F 73 E3  += 075E
1310 7310  E5 AF ED 52 E1 C8 0E 2D CD 06 60 23 C3 10 73 0E  += 0761
1320 7320  5E CD 06 60 CD 55 62 3A 5A 9F B7 CA D8 71 CD 55  += 0834
1330 7330  62 C9 AF 32 61 9F 22 65 9F E5 CD 30 74 DA 46 73  += 081B
1340 7340  DD 36 00 3A DD 23 DD 36 00 20 DD 23 E1 CD 5C 73  += 06FD
1350 7350  3A 61 9F B7 CA 5B 73 2A 65 9F 23 C9 AF 32 64 9F  += 0787
1360 7360  7E 32 63 9F FE CB CA 9B 73 FE DD CA D8 73 FE FD  += 0B3E
1370 7370  CA E1 73 FE ED CA EA 73 FD 21 54 7B 4F 06 00 FD  += 096F
1380 7380  09 FD 7E 00 FE 80 DA 8F 73 CD 63 74 C3 99 73 CD  += 091E
1390 7390  23 74 FD 21 8D 78 CD 5A 78 23 C9 23 3A 64 9F B7  += 075C
13A0 73A0  CA BE 73 7E 32 62 9F 23 7E E6 07 FE 06 CA BE 73  += 0839
13B0 73B0  3E 00 CD 23 74 FD 21 8D 78 CD 5A 78 23 C9 7E 32  += 0700
13C0 73C0  63 9F FD 21 D4 7C 0F 0F 0F E6 1F 4F 06 00 FD 09  += 05FD
13D0 73D0  FD 7E 00 CD 63 74 23 C9 3E 01 32 64 9F 23 C3 60  += 06C5
13E0 73E0  73 3E 02 32 64 9F 23 C3 60 73 23 7E 32 63 9F FE  += 0674
13F0 73F0  C0 D2 0A 74 FE 40 DA 0A 74 FD 21 54 7C D6 40 4F  += 07F9
1400 7400  06 00 FD 09 FD 7E 00 C3 0C 74 3E 00 FE 80 DA 17  += 0677
1410 7410  74 CD 63 74 C3 21 74 CD 23 74 FD 21 8D 78 CD 5A  += 081E
1420 7420  78 23 C9 FE 00 C2 2F 74 F5 3E 01 32 61 9F F1 C9  += 07E7
1430 7430  FD E5 DD E5 FD 21 C9 81 E5 CD B6 0B E1 D2 47 74  += 0AED
1440 7440  FD 21 45 0A CD B6 0B DD E1 DA 60 74 FD 7E 00 E6  += 08C8
1450 7450  7F DD 77 00 FD CB 00 7E DD 23 FD 23 CA 4C 74 AF  += 0872
1460 7460  FD E1 C9 FD 21 A5 79 D6 80 4F 06 00 FD 09 FD 09  += 089A
1470 7470  FD 4E 00 FD 46 01 C5 FD E1 FD 7E 00 FD E5 FD 21  += 09AD
1480 7480  25 79 CD 5A 78 FD E1 DD 36 00 20 DD 23 FD 23 FD  += 086B
1490 7490  7E 00 FE 00 CA A2 74 CD A3 74 FD 23 FD 7E 00 C3  += 089E
14A0 74A0  92 74 C9 FE 01 C2 C0 74 FD 7E 01 FE 00 CA BD 74  += 0939
14B0 74B0  DD 77 00 DD 23 FD 23 FD 7E 01 C3 AB 74 C3 CB 76  += 08D6
14C0 74C0  FE 02 C2 CF 74 23 5E 23 56 CD FE 77 C3 CB 76 FE  += 0943
14D0 74D0  03 C2 DC 74 23 7E CD 20 78 C3 CB 76 FE 04 C2 F9  += 08DC
14E0 74E0  74 23 5E CB 7B CA ED 74 16 FF C3 EF 74 16 00 13  += 07CA
14F0 74F0  EB 19 EB CD FE 77 C3 CB 76 FE 05 C2 14 75 DD 36  += 0996
1500 7500  00 28 DD 23 23 5E 23 56 CD FE 77 DD 36 00 29 DD  += 067D
1510 7510  23 C3 CB 76 FE 06 C2 22 75 DD 36 00 2C DD 23 C3  += 0786
1520 7520  CB 76 FE 07 C2 35 75 FD E5 FD 21 9E 77 CD 44 77  += 094F
1530 7530  FD E1 C3 CB 76 FE 08 C2 48 75 FD E5 FD 21 AE 77  += 0A8C
1540 7540  CD 44 77 FD E1 C3 CB 76 FE 09 C2 58 75 3A 63 9F  += 093C
1550 7550  E6 07 CD CC 76 C3 CB 76 FE 0A C2 6B 75 3A 63 9F  += 08E6
1560 7560  0F 0F 0F E6 07 CD CC 76 C3 CB 76 FE 0B C2 96 75  += 0803
```

zu Abb. 8.3.5

```
-------------- Seite  8  ------- Datei Assem 2.1 -----------
Rom  Abs                                                    Checksumme
1570 7570  DD 36 00 41 DD 23 DD 36 00 2C DD 23 DD 36 00 28   += 05CE
1580 7580  DD 23 FD E5 FD 21 AE 77 CD 44 77 FD E1 DD 36 00   += 099E
1590 7590  29 DD 23 C3 CB 76 FE 0C C2 C1 75 DD 36 00 28 DD   += 0847
15A0 75A0  23 FD E5 FD 21 AE 77 CD 44 77 FD E1 DD 36 00 29   += 08EA
15B0 75B0  DD 23 DD 36 00 2C DD 23 DD 36 00 41 DD 23 C3 CB   += 0721
15C0 75C0  76 FE 0D C2 DC 75 3A 63 9F 0F 0F 0F E6 07 FD E5   += 07CC
15D0 75D0  FD 21 DE 77 CD 5A 78 FD E1 C3 CB 76 FE 0E C2 F7   += 0AB9
15E0 75E0  75 3A 63 9F 0F 0F 0F E6 03 FD E5 FD 21 DE 77 CD   += 07E9
15F0 75F0  5A 78 FD E1 C3 CB 76 FE 0F C2 1F 76 3A 64 9F B7   += 090C
1600 7600  C2 09 76 CD 91 77 C3 1C 76 FE 01 C2 14 76 CD 77   += 07FA
1610 7610  77 C3 1C 76 FE 02 C2 1C 76 CD 84 77 C3 CB 76 FE   += 08EA
1620 7620  10 C2 2D 76 DD 36 00 41 DD 23 C3 CB 76 FE 13 C2   += 07A0
1630 7630  47 76 DD 36 00 28 DD 36 01 43 DD 36 02 29 DD 23   += 058D
1640 7640  DD 23 DD 23 C3 CB 76 FE 14 C2 5B 76 DD 36 00 53   += 080F
1650 7650  DD 36 01 50 DD 23 DD 23 C3 CB 76 FE 15 C2 69 76   += 081C
1660 7660  DD 36 00 28 DD 23 C3 CB 76 FE 16 C2 77 76 DD 36   += 0815
1670 7670  00 29 DD 23 C3 CB 76 FE 17 C2 87 76 3A 63 9F E6   += 0823
1680 7680  07 CD E1 76 C3 CB 76 FE 18 C2 9E 76 3A 63 9F 0F   += 0866
1690 7690  0F 0F E6 07 C6 30 DD 77 00 DD 23 C3 CB 76 FE 11   += 0768
16A0 76A0  C2 B7 76 DD 36 00 28 DD 23 23 7E CD 20 78 DD 36   += 0743
16B0 76B0  00 29 DD 23 C3 CB 76 FE 12 C2 CB 76 3A 63 9F 0F   += 078B
16C0 76C0  0F 0F E6 07 C6 30 DD 77 00 DD 23 C9 4F F5 3A 64   += 0700
16D0 76D0  9F B7 CA E0 76 79 FE 06 C2 E0 76 23 7E 32 62 9F   += 08DF
16E0 76E0  F1 FD E5 FD 21 BE 77 4F FE 06 C2 3D 77 3A 64 9F   += 092C
16F0 76F0  FE 01 C2 13 77 DD 36 00 28 DD 23 CD 77 77 DD 36   += 0754
1700 7700  00 2B DD 23 3A 62 9F CD 20 78 DD 36 00 29 DD 23   += 0607
1710 7710  C3 3A 77 FE 02 C2 36 77 DD 36 00 28 DD 23 CD 84   += 076F
1720 7720  77 DD 36 00 2B DD 23 3A 62 9F CD 20 78 DD 36 00   += 0668
1730 7730  29 DD 23 C3 3A 77 79 CD 5A 78 C3 41 77 79 CD 5A   += 07D0
1740 7740  78 FD E1 C9 3A 63 9F 0F 0F 0F 0F E6 03 4F FE 02   += 06CF
1750 7750  C2 72 77 3A 64 9F FE 01 C2 61 77 CD 77 77 C3 6F   += 086E
1760 7760  77 FE 02 C2 6C 77 CD 84 77 C3 6F 77 CD 91 77 C3   += 0925
1770 7770  76 77 79 CD 5A 78 C9 DD 36 00 49 DD 36 01 58 DD   += 0773
1780 7780  23 DD 23 C9 DD 36 00 49 DD 36 01 59 DD 23 DD 23   += 06B5
1790 7790  C9 DD 36 00 48 DD 36 01 4C DD 23 DD 23 C9 42 43   += 06D2
17A0 77A0  20 20 44 45 20 20 48 4C 20 20 53 50 20 20 42 43   += 0345
17B0 77B0  20 20 44 45 20 20 48 4C 20 20 41 46 20 20 42 20   += 0306
17C0 77C0  20 20 43 20 20 20 44 20 20 20 45 20 20 20 48 20   += 0294
17D0 77D0  20 20 4C 20 20 20 28 48 4C 29 41 20 20 20 4E 5A   += 031A
17E0 77E0  20 20 5A 20 20 20 4E 43 20 20 43 20 20 20 50 4F   += 030D
17F0 77F0  20 20 50 45 20 20 50 20 20 20 4D 20 20 20 E5 EB   += 0442
1800 7800  CD 30 74 E1 D0 3E 9F BA D2 11 78 DD 36 00 30 DD   += 0834
1810 7810  23 7A CD 38 78 7B CD 38 78 DD 36 00 48 DD 23 C9   += 0736
1820 7820  FE 9F CA 2E 78 DA 2E 78 DD 36 00 30 DD 23 CD 38   += 07D5
1830 7830  78 DD 36 00 48 DD 23 C9 F5 0F 0F 0F 0F CD 41 78   += 0653
1840 7840  F1 E6 0F FE 09 CA 52 78 DA 52 78 D6 0A C6 41 C3   += 08CF
1850 7850  54 78 C6 30 DD 77 00 DD 23 C9 C5 FD E5 4F 06 00   += 07DB
1860 7860  FD 09 FD 09 FD 09 FD 09 06 00 FD 7E 00 FE 20 CA   += 0781
1870 7870  89 78 78 FE 04 CA 89 78 FD 7E 00 DD 77 00 DD 23   += 0815
```

zu Abb. 8.3.5

```
-------------- Seite  9  ------- Datei Assem 2.1 -----------
Rom  Abs                                                          Checksumme
1880 7880  FD 23 04 FD 7E 00 C3 6D 78 FD E1 C1 C9 49 4C 4C   += 0890
1890 7890  20 4E 4F 50 20 52 4C 43 41 52 52 43 41 52 4C 41   += 0456
18A0 78A0  20 52 52 41 20 44 41 41 20 43 50 4C 20 53 43 46   += 03E6
18B0 78B0  20 43 43 46 20 45 58 58 20 4E 45 47 20 52 45 54   += 0406
18C0 78C0  4E 49 4D 30 20 52 45 54 49 49 4D 31 20 49 4D 32   += 0417
18D0 78D0  20 52 52 44 20 52 4C 44 20 4C 44 49 20 43 50 49   += 03FF
18E0 78E0  20 49 4E 49 20 4F 55 54 49 4C 44 44 20 43 50 44   += 042C
18F0 78F0  20 49 4E 44 20 4F 55 54 44 4C 44 49 52 43 50 49   += 045E
1900 7900  52 49 4E 49 52 4F 54 49 52 4C 44 44 52 43 50 44   += 04BF
1910 7910  52 49 4E 44 52 4F 54 44 52 48 41 4C 54 45 49 20   += 048F
1920 7920  20 44 49 20 20 4C 44 20 20 49 4E 43 20 44 45 43   += 0383
1930 7930  20 45 58 20 20 41 44 44 20 44 4A 4E 5A 4A 52 20   += 03D8
1940 7940  20 41 44 43 20 53 55 42 20 53 42 43 20 41 4E 44   += 03DD
1950 7950  20 58 4F 52 20 4F 52 20 20 43 50 20 20 52 45 54   += 03D8
1960 7960  20 50 4F 50 20 4A 50 20 20 43 41 4C 4C 50 55 53   += 041D
1970 7970  48 52 53 54 20 49 4E 20 20 4F 55 54 20 52 4C 43   += 0431
1980 7980  20 52 52 43 20 52 4C 20 20 52 52 20 20 53 4C 41   += 03C9
1990 7990  20 53 52 41 20 53 52 4C 20 42 49 54 20 52 45 53   += 0420
19A0 79A0  20 53 45 54 20 2F 7A 34 7A 37 7A 3A 7A 3D 7A 40   += 04DF
19B0 79B0  7A 45 7A 4E 7A 53 7A 56 7A 59 7A 5C 7A 5F 7A 64   += 0684
19C0 79C0  7A 69 7A 6E 7A 73 7A 78 7A 7D 7A 82 7A 87 7A 8A   += 07A2
19D0 79D0  7A 8F 7A 92 7A 95 7A 98 7A 9B 7A 9D 7A A0 7A A3   += 0899
19E0 79E0  7A A6 7A A9 7A AE 7A B1 7A B6 7A BD 7A C5 7A CA   += 0980
19F0 79F0  7A CF 7A D4 7A D9 7A DC 7A E1 7A E4 7A E7 7A EA   += 0ABE
1A00 7A00  7A ED 7A F0 7A F5 7A FA 7A FF 7A 04 7B 09 7B 0E   += 08B8
1A10 7A10  7B 14 7B 19 7B 1E 7B 24 7B 2A 7B 30 7B 33 7B 36   += 050A
1A20 7A20  7B 39 7B 3C 7B 3F 7B 42 7B 45 7B 4A 7B 4F 7B 00   += 05AC
1A30 7A30  07 06 02 00 00 0C 00 01 07 00 01 0A 00 02 0A 00   += 003A
1A40 7A40  00 0A 06 03 00 03 01 41 46 2C 41 46 27 00 04 0F   += 018B
1A50 7A50  06 07 00 00 0B 00 02 07 00 05 04 00 06 04 00 06   += 003A
1A60 7A60  0E 06 04 00 00 05 06 0F 00 00 0F 06 05 00 00 05   += 0051
1A70 7A70  06 10 00 00 10 06 05 00 00 0A 06 09 00 04 10 06   += 0064
1A80 7A80  09 00 07 10 06 09 00 08 09 00 09 10 06 09 00 0A   += 0072
1A90 7A90  09 00 0B 09 00 0C 09 00 0D 09 00 0E 00 0E 0D 00   += 0071
1AA0 7AA0  12 08 00 0F 08 00 11 02 00 11 0D 06 02 00 10 02   += 007C
1AB0 7AB0  00 10 0D 06 02 00 03 15 14 16 06 0F 00 03 01 44   += 00C4
1AC0 7AC0  45 2C 48 4C 00 14 10 06 11 00 15 11 06 10 00 04   += 0180
1AD0 7AD0  10 06 03 00 07 10 06 03 00 08 03 00 09 10 06 03   += 0066
1AE0 7AE0  00 0A 03 00 0B 03 00 0C 03 00 0D 03 00 13 12 00   += 005F
1AF0 7AF0  10 15 0F 16 00 00 14 06 0F 00 14 0A 06 13 00 15   += 00BF
1B00 7B00  13 06 0A 00 09 0F 06 07 00 00 05 06 07 00 00 01   += 005B
1B10 7B10  49 2C 41 00 07 0F 06 07 00 00 07 06 05 00 00 01   += 00EC
1B20 7B20  52 2C 41 00 00 01 41 2C 49 00 00 01 41 2C 52 00   += 0236
1B30 7B30  16 17 00 17 17 00 18 17 00 19 17 00 1A 17 00 1B   += 0106
1B40 7B40  17 00 1C 17 00 1D 18 06 17 00 1E 18 06 17 00 1F   += 010E
1B50 7B50  18 06 17 00 01 80 81 82 83 84 85 02 86 87 88 89   += 0565
1B60 7B60  83 84 85 03 8A 80 81 82 83 84 85 04 8B 87 88 89   += 074F
1B70 7B70  83 84 85 05 8C 80 8D 82 83 84 85 06 8C 87 8E 89   += 0768
1B80 7B80  83 84 85 07 8C 80 8F 82 83 84 85 08 8C 87 90 89   += 0770
```

zu Abb. 8.3.5

```
------------- Seite  10  ------- Datei Assem 2.1 -----------
Rom  Abs                                                          Checksumme
1B90 7B90  83 84 85 09 91 91 91 91 91 91 91 91 91 91 91 91   += 0861
1BA0 7BA0  91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   += 0910
1BB0 7BB0  91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   += 0910
1BC0 7BC0  91 91 91 91 91 91 91 91 91 91 23 91 91 91 91 91   += 08A2
1BD0 7BD0  91 91 91 91 92 92 92 92 92 92 92 92 93 93 93 93   += 0920
1BE0 7BE0  93 93 93 93 94 94 94 94 94 94 94 94 95 95 95 95   += 0940
1BF0 7BF0  95 95 95 95 96 96 96 96 96 96 96 96 97 97 97 97   += 0960
1C00 7C00  97 97 97 97 98 98 98 98 98 98 98 98 99 99 99 99   += 0980
1C10 7C10  99 99 99 99 9B 9D A1 A0 9F 9C A6 AE 9B 9A A1 00   += 0942
1C20 7C20  9F 9E A7 AE 9B 9D A1 A5 9F 9C A8 AE 9B 0A A1 A4   += 098B
1C30 7C30  9F 00 A9 AE 9B 9D A1 A2 9F 9C AA AE 9B AF A1 A3   += 0992
1C40 7C40  9F 00 AB AE 9B 9D A1 25 9F 9C AC AE 9B B0 A1 24   += 089B
1C50 7C50  9F 00 AD AE B1 B2 B3 B4 0B 0C 0D B5 B1 B2 B6 B7   += 086D
1C60 7C60  00 0E 00 B8 B1 B2 B3 B4 00 00 0F B9 B1 B2 B6 B7   += 0728
1C70 7C70  00 00 10 BA B1 B2 B3 00 00 00 00 11 B1 B2 B6 00   += 050A
1C80 7C80  00 00 00 12 00 00 B3 B4 00 00 00 00 B1 B2 B6 B7   += 0449
1C90 7C90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1CA0 7CA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1CB0 7CB0  00 00 00 00 13 14 15 16 00 00 00 00 17 18 19 1A   += 00B4
1CC0 7CC0  00 00 00 00 1B 1C 1D 1E 00 00 00 00 1F 20 21 22   += 00F4
1CD0 7CD0  00 00 00 00 BB BC BD BE BF C0 00 C1 C2 C2 C2 C2   += 083A
1CE0 7CE0  C2 C2 C2 C2 C3 C3 C3 C3 C3 C3 C3 C3 C4 C4 C4 C4   += 0C30
1CF0 7CF0  C4 C4 C4 C4 7E 23 B7 C8 4F CD 06 60 18 F6 1A 44   += 081E
1D00 7D00  65 62 75 67 67 65 72 20 56 32 2E 31 20 28 43 29   += 049C
1D10 7D10  20 31 39 38 34 20 52 6F 6C 66 2D 44 69 65 74 65   += 04C1
1D20 7D20  72 20 4B 6C 65 69 6E 0D 0A 0D 0A 20 31 3D 41 73   += 03F5
1D30 7D30  73 65 6D 62 6C 65 72 0D 0A 20 32 3D 44 69 73 61   += 0511
1D40 7D40  73 73 65 6D 62 6C 65 72 0D 0A 20 33 3D 53 74 61   += 052C
1D50 7D50  72 74 65 6E 0D 0A 20 34 3D 45 69 6E 7A 65 6C 73   += 053B
1D60 7D60  63 68 72 69 74 74 0D 0A 20 35 3D 50 72 6F 74 6F   += 054B
1D70 7D70  6B 6F 6C 6C 20 61 6E 2F 61 75 73 0D 0A 20 36 3D   += 04C3
1D80 7D80  47 72 75 6E 64 70 72 6F 67 72 61 6D 6D 0D 0A 0D   += 0589
1D90 7D90  0A 45 69 6E 67 61 62 65 20 3E 3E 00 53 74 61 72   += 04EB
1DA0 7DA0  74 2D 41 64 72 65 73 73 65 3A 00 45 6E 64 2D 41   += 0527
1DB0 7DB0  64 72 65 73 73 65 3A 00 CD 2A 60 3E 21 D3 73 AF   += 066B
1DC0 7DC0  D3 72 C9 31 CD 9F CD B8 7D AF 32 4F 9F 32 50 9F   += 089D
1DD0 7DD0  32 51 9F 21 00 00 22 52 9F 21 00 88 22 54 9F 22   += 0436
1DE0 7DE0  67 9F 22 56 9F 3A 4F 9F F5 AF 32 4F 9F 21 FE 7C   += 07A4
1DF0 7DF0  CD F4 7C CD 03 60 4F F5 CD 06 60 F1 4F F1 32 4F   += 0896
1E00 7E00  9F 79 FE 31 C2 12 7E 0E 1A CD 06 60 CD D1 71 C3   += 07C6
1E10 7E10  29 7F FE 32 C2 EF 7E CD 55 62 0E 1A CD 06 60 21   += 0707
1E20 7E20  9C 7D CD F4 7C CD 3D 62 DD 21 00 9F CD 70 61 FD   += 08FA
1E30 7E30  21 00 9F CD 0C 70 CD 55 62 DA 1A 7E 22 67 9F CD   += 06F4
1E40 7E40  55 62 0E 1A CD 06 60 21 AB 7D CD F4 7C CD 3D 62   += 0704
1E50 7E50  DD 21 00 9F CD 70 61 FD 21 00 9F CD 0C 70 CD 55   += 0763
1E60 7E60  62 DA 42 7E 22 5F 9F 0E 1A CD 06 60 06 16 E5 21   += 0599
1E70 7E70  CD 7E E3 C5 CD 3D 62 DD 21 00 9F 2A 67 9F E5 CD   += 08DE
1E80 7E80  B8 61 E1 E5 DD 21 0E 9F CD 32 73 22 67 9F DD 36   += 0837
1E90 7E90  00 00 21 00 9F CD F4 7C CD 55 62 E1 7E FE C9 C2   += 0869
```

zu Abb. 8.3.5

```
-------------- Seite  11  ------- Datei Assem 2.1 -----------
Rom  Abs                                                       Checksumme
1EA0 7EA0  A8 7E CD 55 62 C1 05 C5 C1 05 78 B7 CA B5 7E 78   += 089F
1EB0 7EB0  FE FF C2 B7 7E AF C9 2A 5F 9F ED 5B 67 9F AF ED   += 0A7E
1EC0 7EC0  52 7C B7 F2 CA 7E 3E 03 B7 C9 C3 73 7E FE 03 C2   += 08F7
1ED0 7ED0  DA 7E CD 03 60 3E 03 C3 DD 7E CD 03 60 FE 6D CA   += 084C
1EE0 7EE0  EC 7E FE 4D CA EC 7E FE 03 C2 67 7E C3 29 7F FE   += 09FA
1EF0 7EF0  33 C2 00 7F CD 74 60 CD 86 1B CD B8 7D C3 29 7F   += 07F0
1F00 7F00  FE 34 C2 11 7F CD 74 60 CD 3F 1A CD B8 7D C3 29   += 0839
1F10 7F10  7F FE 35 C2 21 7F 3A 4F 9F EE 01 32 4F 9F C3 29   += 0737
1F20 7F20  7F FE 36 C2 29 7F C3 00 00 C3 E5 7D 00 00 00 00   += 0605
1F30 7F30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1F40 7F40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1F50 7F50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1F60 7F60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1F70 7F70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1F80 7F80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1F90 7F90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1FA0 7FA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1FB0 7FB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1FC0 7FC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1FD0 7FD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1FE0 7FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
1FF0 7FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   += 0000
```

Abb. 8.3.5 Hexdump des Debuggers

8.4 Das ist Gosi

Gosi ist der Sprache Logo entlehnt, und zwar der deutschen IWT-Version. Sie enthält viele Sprachelemente daraus, allerdings keine Listenverarbeitung. Mit Gosi lassen sich alle Merkmale einer höheren Programmiersprache zeigen, wie strukturierte Programmierung, Prozeduren und Parameterkonzepte. Der Übergang von unserer Zeichensprache zu einer höheren Programmiersprache läßt sich durch Gegenüberstellung von Programmbeispielen zeigen.

Gosi ist die Abkürzung für „Graphisch Orientierte Sprache I“. Die Sprache Gosi ist verwandt mit der Sprache Logo, die besonders leicht zu erlernen ist. Gosi läuft auf der SBC2-Karte anstelle des Grundprogramms. Nach dem Einschalten meldet sich Gosi, wie in *Abb. 8.4.1* gezeigt.

Gosi enthält eine Schildkröte, die man bewegen kann, genau wie bei der Zeichensprache. Doch die Befehle sind jetzt einfacher. Man muß keinen Maschinencode mehr eingeben.
Erstes Beispiel:
Für die Vorwärtsbewegung gibt es den Befehl

VORWAERTS

Wieviel geschritten werden soll, wird durch eine Zahl definiert, die man hinter den Befehl tippt (durch ein Leerzeichen trennen).

VORWAERTS 100

Abb. 8.4.1 So meldet sich Gosi

```
GOSI - Graphisch orientierte Sprache I
(C) Muenchen 1984 Rolf-Dieter Klein Vers 1.1
"Ein Hauch von LOGO ..."
```

Wenn man dann die Taste CR betätigt, so wird der Befehl ausgeführt, und es erscheint eine Linie auf dem Bildschirm. Bei der Zeichensprache mußte man die Sequenz:

21#100.W
CD SCHREITE

programmieren, um das gleiche Ergebnis zu erhalten. Die höhere Programmiersprache ist also in der Lage, durch einen Befehl die Ausführung mehrerer Maschinenbefehle zu erzwingen. Der Übersetzer, d. h. das Programm, das die Eingabe in eine Befehlssequenz umsetzt, kann je nach Konzeption unterschiedlich arbeiten. Beim sogenannten Interpreter wird ein Befehl eingelesen, wie z. B. VORWAERTS 100 und dann sofort ausgeführt; dann wird der nächste Befehl eingelesen, wieder analysiert und ausgeführt.

Bei einem sogenannten Compiler wird erst einmal alles eingelesen und dann zunächst komplett in eine Maschinensequenz umgesetzt. Erst, wenn das ganze Programm fertig als Maschinencode dasteht, wird es ausgeführt.

Die Progammiersprache Gosi ist ein Interpreter, denn ein Interpreter besitzt den Vorteil, daß man auch einmal einen einzelnen Befehl per Tastatur eingeben kann und sofort das Ergebnis auf dem Bildschirm sieht. Bei einem reinen Compiler müßte man zunächst das ganze Programm eintippen und könnte erst nach der vollständigen Übersetzung das Ergebnis betrachten. Interpreter arbeiten dafür aber im allgemeinen ein Programm langsamer ab als ein von Compilern erzeugtes Maschinenprogramm, da sie alle Befehle immer wieder von neuem übersetzen müssen.

Einige Befehle von Gosi

Neben dem VORWAERTS-Befehl gibt es auch: RUECKWAERTS

Alle Befehle kann man auch abkürzen:

VORWAERTS = VW
RUECKWAERTS = RW

Zum Drehen der Schildkröte gibt es:

LINKS = LI
RECHTS = RE

dann

STIFTAB = SA
STIFTHOCH = SH

und

SCHR16TEL = entsprechend der Zeichensprache.

Abb. 8.4.2 zeigt ein Programmbeispiel: ein Beispiel zum Übersetzungsvorgang. Man könnte folgendes Gosi-Programm:

VW 20 RE 40 LI 50 RW 10

in den nachfolgenden Code umsetzen:

```
21 #20.W
CD SCHREITE
21 -#40.W
CD DREHE
21 #50.W
CD DREHE
21 -#10.W
CD SCHREITE
```

Wichtig ist, daß man in Gosi auch neue Befehle definieren kann. Das geschieht mit dem Befehl:

LERNE

Danach folgt der Name des zu lernenden Programms. Danach die Befehle des Programmes bis zum Befehl ENDE, der angibt, daß hier die Definition endet.

Als Konstruktion für Schleifen gibt es den Befehl:

WIEDERHOLE oder einfach WH

Danach folgt die Anzahl der Wiederholungen und dann in eckigen Klammern die Befehle, die wiederholt werden sollen. So zeigt *Abb. 8.4.3* die Definition eines Quadrats und *Abb. 8.4.4* die Definition eines Kreises.

Jetzt kommt etwas Neues

Die Verwendung von Parametern. Wenn man Quadrate oder Kreise haben will, die unterschiedlich groß sind, so muß man dem Programm einen Parameter, die Größen mitgeben. Dies geschieht z. B. in *Abb. 8.4.5*.

Der Wert „:n“ in der Zeile mit dem Lerne-Befehl definiert einen Parameter mit dem Namen „n“. Der Doppelpunkt wird als Erkennung benutzt. In dem Programm kann man den Parameter „n“ so verwenden, als sei es eine Zahl. Wenn man dann das Programm aufruft, so muß man eine Zahl hinter dem Befehl mit angeben.

Beispiel: Ecken 3

Die Zahl 3 wird dann dem Parameter „n“ zugeordnet und steht in dem Programm als solcher zur Verfügung. In dem Programm ECKEN geschieht aber doch was anderes. Am Schluß steht der

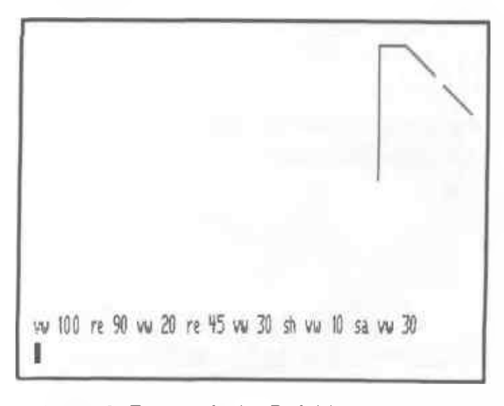


Abb. 8.4.2 Eine einfache Befehlssequenz in Gosi

```
lerne quadrat
wh 4[vw 100 re 90]
ende
```

Abb. 8.4.3 Ein Quadrat wird definiert

```
lerne kreis
wh 36 [vw 10 re 10]
ende
```

Abb. 8.4.4 Ein Kreis wird definiert

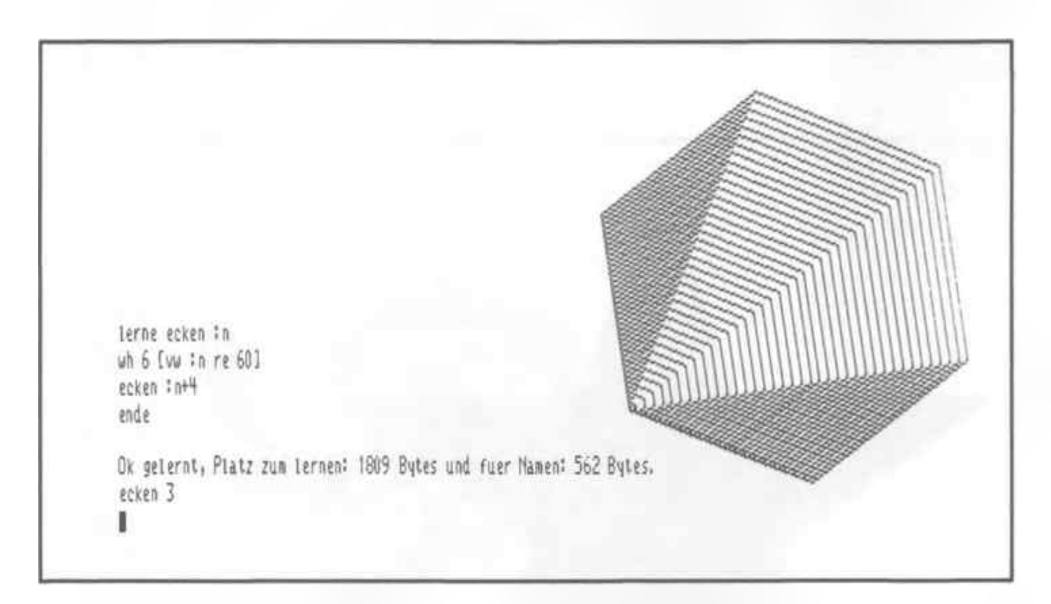


Abb. 8.4.5 Die Verwendung von Parametern

Befehl „ecken:n+4“. Das bedeutet, das Programm ruft sich selbst erneut auf, verwendet aber diesmal als Parameter den um vier erhöhten Wert von „n“ beim nächsten Durchlauf.

Dieses Programm wird nie enden. Man nennt diese spezielle Aufrufform auch „tail recursion“, in der Wirkung kommt sie einem Sprung gleich, nur daß auch Parameter verwendet werden.

Wenn man das Programm anhalten will, so muß man die Tasten „CTRL“ und gleichzeitig „S“ drücken. Will man das Programm weiterlaufen lassen, so drückt man „CTRL“ und gleichzeitig

```
lerne vspirale :laenge :winkel
vw :laenge re :winkel
vspirale (:laenge+3) :winkel
ende
```

Abb. 8.4.6 VSPIRALE – Ein Programm mit vielen Gesichtern

„Q". Will man das Programm abbrechen, so muß man zuerst „CTRL" und „S" gedrückt haben und dann „CTRL" und „C" drücken.

Man kann auch mehrere Parameter angeben, wie in *Abb. 8.4.6*. Es ergeben sich dann Figuren, wie *Abb. 8.4.7*, *Abb. 8.4.8*, *Abb. 8.4.9* und *Abb. 8.4.10*. Jedes dieser Bilder zeigt eine besondere Form des Programms „VSPIRALE". Man kann es einmal mit ähnlichen Winkeln beim Aufruf probieren und wird diese Grundtypen immer wieder erkennen.

Da Gosi sehr umfangreich ist, sei hier auf die dem EPROM-Satz beigefügte Beschreibung sowie auf das Buch „Einführung in Logo" von Harald Abelson, erschienen im IWT-Verlag, München, verwiesen, in dem es viele ausführliche Beispiele gibt. Abschließend noch ein paar besondere Anwendungen von Gosi, die in den genannten Werken nicht vorkommen.

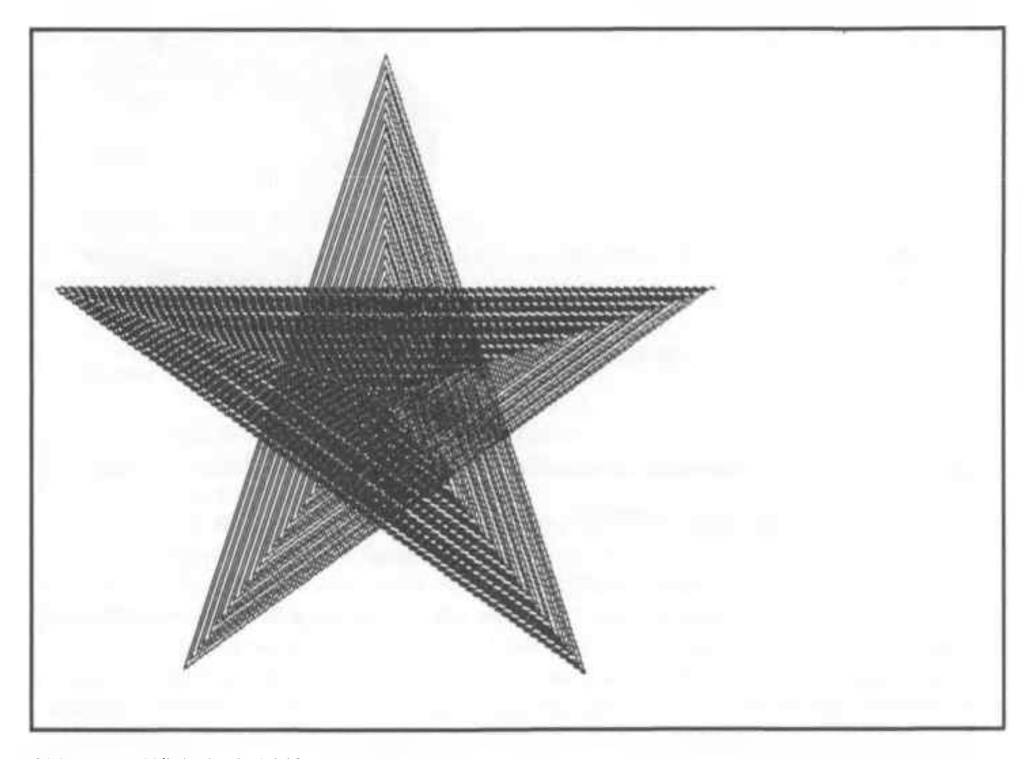


Abb. 8.4.7 Winkel mit 144°

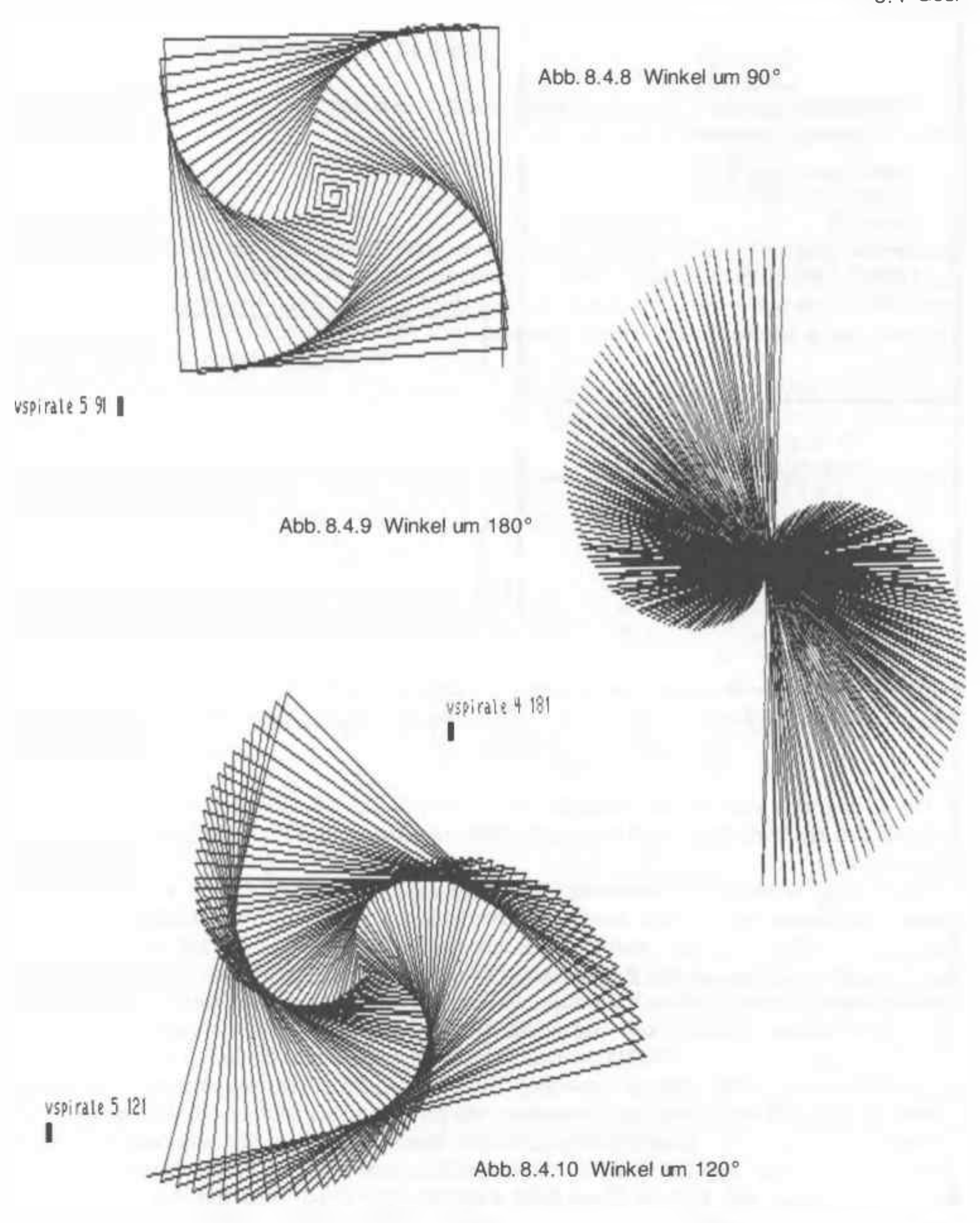


Abb. 8.4.8 Winkel um 90°

Abb. 8.4.9 Winkel um 180°

Abb. 8.4.10 Winkel um 120°

Der Anschluß eines Lichtgriffels

Die Baugruppe GDP64 ist für den Anschluß eines Lichtgriffels vorbereitet. Hier ein Beispiel, wie man den Lichtgriffel mit Gosi bedienen kann. Gosi eignet sich nämlich nicht nur zur Erstellung von Grafiken, sondern auch hervorragend für Steuerungen.

```
lerne lichtgriffelxy
solange (:port 240)&1<>0 [seite 0 0]
port 112 8
solange (:port 112)&1=0 [seite 0 0]
setze "x (:port 124)*2  setze "y (:port 125)*2
ende
```

Abb. 8.4.11 Lichtgriffel-Programm

```
lerne testgriffel
lichtgriffelxy
dr :x dr [ ] dz :y
testgriffel
ende
```

Abb. 8.4.12 Testprogramm

```
88 142
96 182
120 182
184 201
240 201
336 192
432 96
```

Abb. 8.4.13 Testausgabe

Der Lichtgriffel wird an den Anschluß LPCK, PIN 21 des EF 9366 angeschlossen. Dort befindet sich ein Widerstand 330 Ω, der nach Masse geschaltet ist. Den Widerstand kann man dann entfernen.

Der Lichtgriffel muß im Normalzustand 0 V führen und einen kurzen Puls nach + 5 V liefern, wenn er beleuchtet wird. Ferner wird eine Taste benötigt, die meist in den Lichtgriffel mit eingebaut ist. Diese Taste wird betätigt, wenn man den Lichtgriffel auf den Schirm drückt. Die Taste soll angeben, wann man die Position erreicht hat, die angemerkt werden soll. Die Taste wird mit Hilfe eines Ports auf der IOE-Karte angeschlossen. Dazu wird die Taste in Bit 0 des 74LS245 (B1) angeschlossen. Die IOE-Karte wird auf die Adresse Fx gelegt, also alle Brücken (4, 5, 6, 7) werden offengelassen. Der 74LS245 ist dann auf der Adresse F0 oder dezimal 240 erreichbar. Man kann auch eine andere Adresse verwenden, wenn man das Programm entsprechend umstellt.

Abb. 8.4.11 zeigt das Lichtgriffel-Programm. Mit der Anweisung SOLANGE wird zunächst gewartet bis die Taste des Lichtgriffels gedrückt wird. Dann wird auf Port 112 (sedezimal 70) der Befehl 8 ausgegeben, der den Bildschirm kurz weiß leuchten läßt, und zwar solange, bis der Lichtgriffeleingang des EF9366 einen Puls entdeckt hat. Dann wird mit den Befehlen ‚SETZE"X(. . . .)‘ der eingelesene Wert an die Variable X übergeben. Die Variable "X steht auf der linken Seite mit einem Anführungszeichen, da ihr ein Wert zugewiesen wird, während sie rechts mit einem Doppelpunkt versehen ist, da ein Wert aus der Variablen genommen wird. Die Variable "Y wird entsprechend belegt.

Abb. 8.4.12 zeigt ein kleines Testprogramm, mit dem man die Eingabe prüfen kann. Es liefert als Ergebnis zum Beispiel die Koordinaten *(Abb. 8.4.13)*, wenn man den Lichtgriffel auf den Bildschirm drückt und damit die Taste betätigt.

Abb. 8.4.14 Das Zeichen-Programm

```
lerne zeichne
lichtgriffelxy
stifthoch aufxy :x :y stiftab
wiederhole 6 [vorwaerts 4 rechts 60]
zeichne
ende
```

Nun kann man ein Zeichenprogramm schreiben. *Abb. 8.4.14* zeigt ein Beispiel. Ein kleines Sechseck wird an der Position ausgegeben, bei der man die Taste am Lichtgriffel gedrückt hat. *Abb. 8.4.15* zeigt eine damit erstellte Zeichnung. Aufgrund der Arbeitsweise des EF9366 kann man in horizontaler Richtung aber nur jeden achten Punkt erreichen, während in der vertikalen Richtung jeder Punkt erreichbar ist.

Auch eine Menüsteuerung ist möglich. *Abb. 8.4.16* zeigt ein Beispiel. Man tippt mit dem Lichtgriffel einen Menüpunkt an und erhält entweder KREIS oder QUADRAT. Die beiden

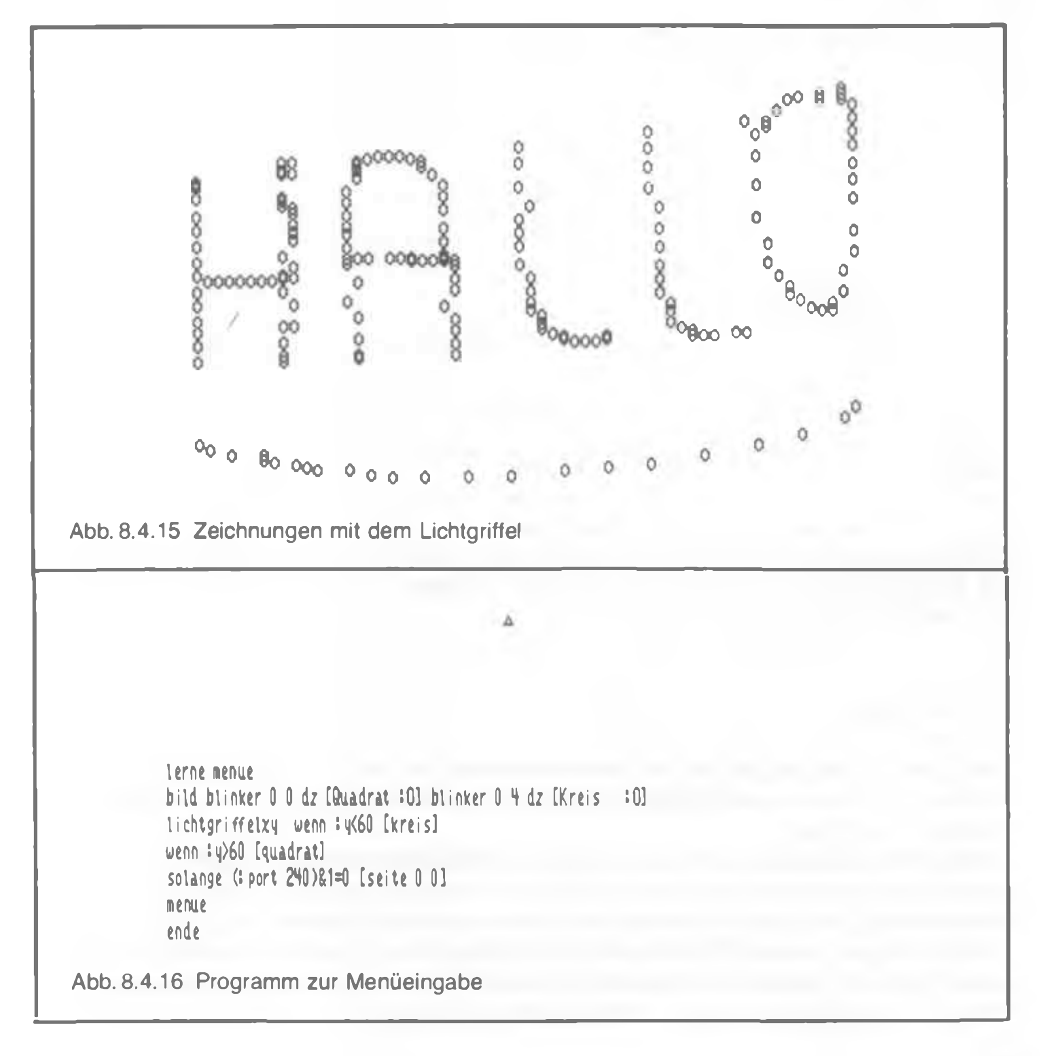


Abb. 8.4.15 Zeichnungen mit dem Lichtgriffel

```
lerne menue
bild blinker 0 0 dz [Quadrat :0] blinker 0 4 dz [Kreis  :0]
lichtgriffelxy  wenn :y<60 [kreis]
wenn :y>60 [quadrat]
solange (:port 240)&1=0 [seite 0 0]
menue
ende
```

Abb. 8.4.16 Programm zur Menüeingabe

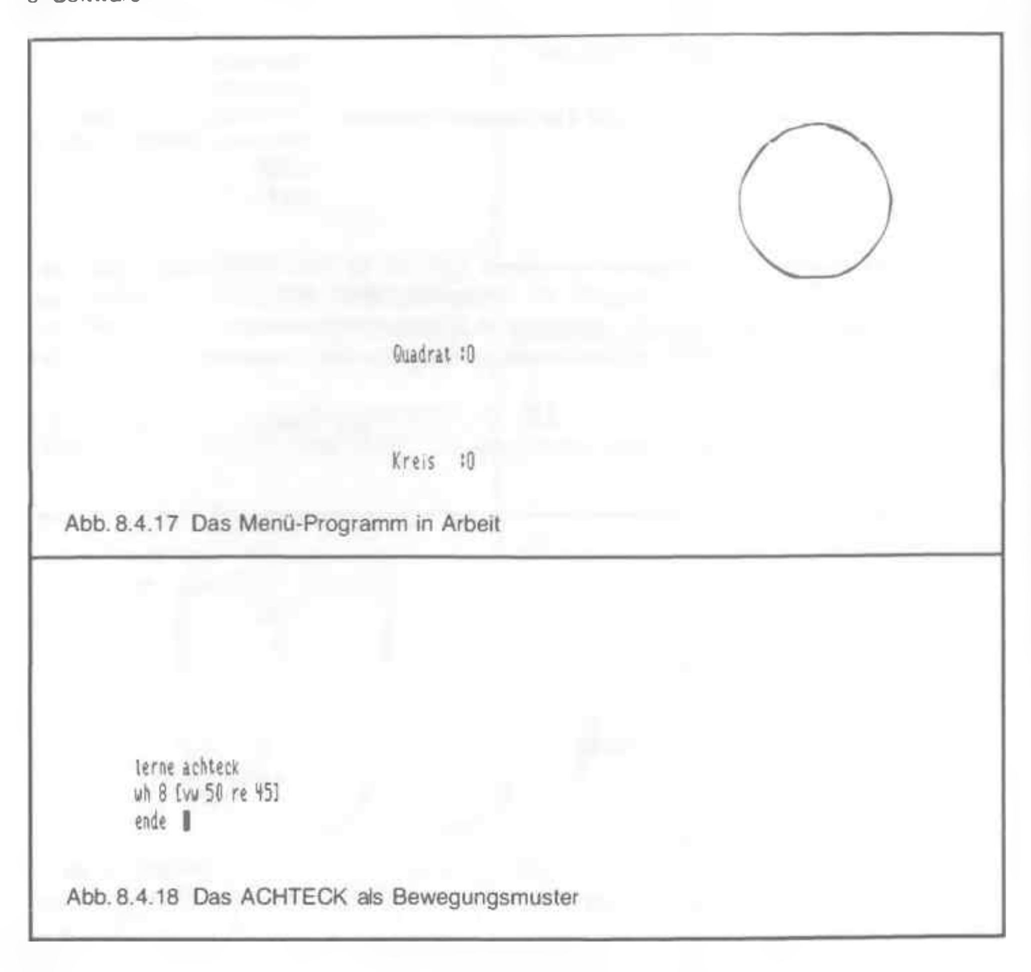


Abb. 8.4.17 Das Menü-Programm in Arbeit

Abb. 8.4.18 Das ACHTECK als Bewegungsmuster

Programme müssen zuvor natürlich auch eingegeben worden sein. *Abb. 8.4.17* zeigt das Ergebnis bei KREIS. Der KREIS oder das QUADRAT erscheinen dabei solange, bis die Taste am Lichtgriffel wieder losgelassen wird.

Bewegte Grafik

Man kann auch Bilder über den Bildschirm bewegen. *Abb. 8.4.18* zeigt das Programm für ein Achteck, das über den Bildschirm bewegt werden soll. In *Abb. 8.4.19* ist das BEWEGE-Programm abgebildet.

Um die Bewegung störungsfrei zu gestalten, wird eine besondere Technik verwendet. Das Bild wird zunächst unsichtbar auf einer Bildebene gezeichnet. Erst danach wird die Bildebene umgeschaltet. Vor dem Zeichnen wird eine eventuell noch vorhandene alte Figur gelöscht.

Als Parameter wird dem Unterprogramm die Anzahl der Bildpunkte zwischen zwei Bewegungsvorgängen gegeben. Damit läßt sich die Geschwindigkeit der Bewegung steuern. *Abb. 8.4.20* zeigt einen Ausschnitt aus der Bewegungsphase.

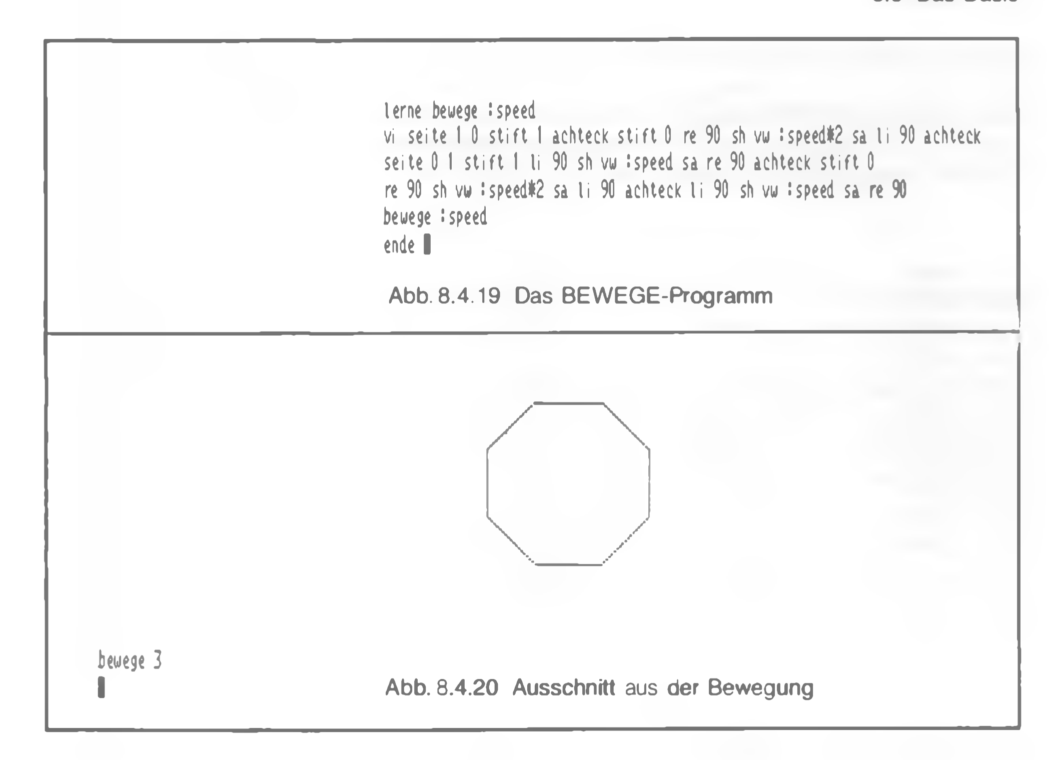


Abb. 8.4.19 Das BEWEGE-Programm

Abb. 8.4.20 Ausschnitt aus der Bewegung

8.5 Das Basic (Dr. Hans Hehl)

8 KByte für den NDR-Klein-Computer

Puristen werden an dieser Stelle die Nase rümpfen, weil wir hier den Befehlssatz eines Basics besprechen. Da die Welt aber nun einmal nicht immer nach den Vorstellungen von Puristen geordnet ist, wollen wir hier nicht die Augen verschließen und so tun, als sei Basic nicht in der Welt. Heute kann jeder Schüler mehr oder weniger gut Programme in Basic schreiben und sie auf Basic-Computern ablaufen lassen. Vielleicht reizt es solche Kenner, auch die anderen Vorzüge des NDR-Klein-Systems kennenzulernen, wenn sie wissen, „der Computer kann auch Basic". Außerdem wird auch der völlig im Sinne der reinen Lehre erzogene Novize irgendwann auf Basic-Programmierer stoßen. Es ist dann gut, wenn er mitreden kann.

Das 8-KByte-Basic für den NDR-Klein-Computer entspricht in etwa dem Microsoft-Standard. Es wird in zwei EPROMs 2732A geliefert, die anstelle der beiden Grundsoftware-EPROMs auf der SBC-2-Platine eingesetzt werden. Da der Interpreter auch den Grafikprozessor steuern sollte, mußten aufgrund des knappen Speicherplatzes Abstriche am Bedienungskomfort gemacht werden.

Sedezimalzahlen („Hexzahlen") werden in diesem Text durch ein nachgestelltes „H" gekennzeichnet. Basic-Befehle werden in der Beschreibung groß geschrieben, dürfen jedoch beliebig in Groß- oder Kleinbuchstaben eingegeben werden. Das Betätigen der Return-Taste wird durch <CR> angedeutet. <CTRL-Q> soll das Drücken der Control-Taste und das gleichzeitige Drücken der Q-Taste symbolisieren. Der Interpreter benötigt zusätzlich Speicherplatz bis zur Adresse 88C4H, ab 88C5H beginnt der Programmspeicher.

Die alphabetische Liste der Basic-Befehle

Die bei manchen Befehlen angegebene Klammer soll darauf hinweisen, daß dem Basic-Befehl ein Ausdruck in Klammern folgen muß, sonst erfolgt die Fehlermeldung „?SN Fehler“ (SYNTAX ERROR), was die falsche Eingabe eines Basic-Befehls signalisiert.

1) ABS(
2) AND
3) ASC(
4) ATN(
5) CALL
6) CHR$(
7) CLEAR
8) CLRS
9) CONT
10) COS(
11) CSAVE
12) CLOAD
13) DATA
14) DEF FN
15) DIM
16) DRAWTO
17) END
18) EXP(
19) FRE(
20) FOR
21) GOSUB
22) GOTO
23) HEX(
24) IF
25) INP(
26) INPUT
27) INT(
28) LEFT$(
29) LEN(
30) LET
31) LIST
32) LLIST
33) LOG(
34) LPRINT
35) MID$(
36) MOVETO
37) NEW
38) NEXT
39) NOT
40) NULL
41) ON
42) OR
43) OUT
44) PAGE
45) PEEK(
46) POKE
47) POS(
48) PRINT
49) READ
50) REM
51) RESTORE
52) RETURN
53) RIGHT$(
54) RND(
55) RUN
56) SGN(
57) SIN(
58) SPC(
59) SQR(
60) STR$(
61) STOP
62) TAB(
63) TAN(
64) USR(
65) VAL(
66) WAIT
67) ? PRINT

Zum Start des Basic-Interpreters

Nach dem Einschalten des Rechners meldet sich der Interpreter mit einem Fragezeichen. Dann muß C (Großbuchstabe!, C für „Cold Start“) getippt werden, worauf folgende Meldung auf dem Bildschirm erscheint:

```
8 K Basic 1.3
RDK 83
o. k.
>
```

Wichtig: Nur „C“ erweckt den Computer zum Leben! Wenn der Computer „schon im Laufen ist“, dann können Sie nach Drücken der Reset-Taste (SBC-2-Platine) mit der Eingabe von „W“ (für Warmstart) den Interpreter starten, ohne Ihr Programm im Speicher zu zerstören. Meldung: „o.k.“ und „>“. Beim Warmstart umgeht der Interpreter die Initialisierungsroutinen.

Die Kontrollfunktionen

Während der Programmausführung fragt der Interpreter ständig die Tastatur ab. Bei Eingabe von <CTRL-S> stoppt die weitere Programmausführung sofort. Mit <CTRL-Q> wird dann wieder gestartet. Dies gilt auch für den Befehl LIST. Wird während der Programmausführung die Escape-Taste <ESC> betätigt, bewirkt dies eine Unterbrechung nach dem gerade ausgeführten Befehl sowie die Meldung „abgebrochen in Zeile XY“. Wie auch bei dem Befehl STOP kann man jetzt die Variablen inspizieren, modifizieren und das Programm mit dem Befehl CONT fortsetzen, sofern es nicht verändert wurde.

Ein aktiver INPUT-Befehl kann nicht mit <ESC> unterbrochen werden, sondern nur durch die Betätigung der Return-Taste; der Interpreter geht daraufhin in die Basic-Anweisungsebene zurück (o.k.-Meldung), kann aber das Programm (per CONT) mit dem INPUT-Befehl fortsetzen, sofern es zwischenzeitlich nicht modifiziert worden ist.

Variablen

Basic läßt die Verarbeitung variabler Größen zu. Eine Rechenvorschrift kann zum Beispiel in Buchstaben-Form angegeben werden (c = a + b) und für die Variablen werden je nach Bedarf unterschiedliche aktuelle Werte eingesetzt.

Basic kann dabei nicht nur Zahlen als variable Größen verarbeiten, sondern auch Buchstaben und Texte (sogenannte „Strings"). Variablen für Zahlen werden durch eine maximal zweistellige, alphanumerische Zeichenkombination (alphanumerisch: aus Ziffern und Buchstaben zusammengesetzt) bezeichnet, wobei an erster Stelle immer ein Buchstabe stehen muß. String-Variablen (maximal zulässige Länge: 255 Zeichen bei entsprechender Speicherplatzreservierung) werden dadurch gekennzeichnet, daß an ihre (höchstens zweistellige) Abkürzung ein „$" angehängt wird. Es ist zulässig, in einem Programm gleichzeitig die numerische Variable „A" und die Stringvariable „A$" zu benutzen, unser Basic kann sie auseinanderhalten.

Die Zahlen

Die interne Zahlendarstellung erfolgt mit sechs Stellen, plus Vorzeichen, plus zweistelligem, vorzeichenbehafteten Exponenten. Ausgegeben werden fünf gültige Stellen, die sechste wird gerundet. Daher ergibt die Eingabe von PRINT 123.456 <CR> die Anzeige „123.45" und die Eingabe von PRINT 123.456 + 123.456 <CR> die Anzeige „246.91". Allerdings erhält man bei Eingabe von PRINT 1 ↑ 100000 <CR> dann auch wirklich den Wert 1.

Vor Zahlen wird ein positives Vorzeichen nicht dargestellt, sondern an dessen Stelle ein Leerzeichen ausgegeben. Es können Zahlen im Bereich von ca. − 5E − 39 bis ca. + 5E + 38 verarbeitet werden; diese scheinbar willkürlichen Grenzbereiche resultieren aus der internen binären Darstellungsform der Zahlen (drei Bytes für Mantisse und Vorzeichen, ein viertes Byte für den Exponenten plus Vorzeichen). Bei Gleitkommazahlen werden die Nachkommastellen nicht durch ein Komma, sondern durch einen Punkt abgetrennt. Das Komma ist in Basic dafür reserviert, zwei voneinander unabhängige Größen (z. B. Variablen in einer Aufzählung) gegeneinander abzugrenzen.

Bei Winkelfunktionen muß der Winkel in Bogenmaß (rad) angegeben werden. Die Umrechnung erfolgt durch Division des Winkelwertes (Grad) mit 180 und Multiplikation mit der Zahl PI (3.1415).

Operatoren

Operatoren nennt man alle die Zeichen, die veranlassen, daß eine Rechenart oder ein Vergleich durchgeführt wird. Operatoren möchten Zahlen oder Strings als Eingabe haben, operieren damit und werfen das Ergebnis als Zahl oder String aus. Einige Operatoren weichen in ihrer Schreibweise von der gewohnten algebraischen Notation ab, und durch den Einsatz von Klammern werden bestimmte Ausdrücke (z. B. im Nenner oder unter einer Wurzel stehende) zusammengefaßt.

+	Addition von Variablen; Strings können durch Addition aneinandergereiht werden
−	Subtraktion von (numerischen) Variablen
*	Multiplikation von (numerischen) Variablen
/	Division von (numerischen) Variablen
=	hat außer der arithmetischen Bedeutung (Gleichheitszeichen) in Basic noch die Zuweisungsfunktion, d. h. einer links vom Gleichheitszeichen stehenden Variablen wird die rechts stehende Zahl, der Inhalt der rechtsstehenden Variablen, der Wert einer Funktion oder die Zeichenkette zugewiesen
<	Vergleichsoperator für „kleiner als"
<=	Vergleichsoperator für „kleiner als oder gleich" (Reihenfolge der beiden Operatoren beliebig)
>	Vergleichsoperator für „größer als"
>=	Vergleichsoperator für „größer als oder gleich" (Reihenfolge der beiden Operatoren beliebig)
<>	Operator für „ungleich"
E	definiert die nachgestellte Zahl (keine Variable!) als Exponent zur Basis Zehn; kann nicht allein, sondern nur in Verbindung mit einer vorangestellten Zahl (nicht Variablen!) verwendet werden BEI EINGABEN MUSS DAS EXPONENTEN-E GROSS GESCHRIEBEN WERDEN!
↑	definiert die nachgestellte Variable als Exponenten zur vorangestellten Basis
PI	(Kreiszahl) ist nicht fest in Basic gespeichert und muß bei Bedarf definiert werden: PI = 3.1415

Die Befehle und Funktionen

ABS(X)
Bildet den Absolutwert von X.

PRINT ABS(−523) <CR> Anzeige: 523
PRINT ABS(−5.1234*10 ↑ 4) <CR> Anzeige: 51234

X AND Y
Bildet auf Maschinen-Ebene bitweise die UND-Verknüpfung aus dem binären Äquivalent von X und Y, d. h. im Ergebniswort steht nur an der Stelle eine 1, an der die korrespondierenden Bits in beiden Operanden X und Y eine 1 hatten.

PRINT 101 AND 95 <CR> Anzeige: 69

ASC ("X") oder **ASC (A$)**
Erzeugt den zum ASCII-Zeichen "X" gehörenden Binärcode und gibt diesen dezimal aus.

PRINT ASC("H") <CR> Anzeige: 72
PRINT ASC("Meier") <CR> Anzeige: 77

ATN (X)
Bildet den Arkustangens vom Argument X. Das Ergebnis wird im Bogenmaß angegeben.

PRINT ATN(2) <CR> Anzeige: 1.1071

CALL X
Springt in das bei der Adresse X beginnende Unterprogramm in Maschinensprache, das mit dem Z80-Befehl „Return" (C9H) abgeschlossen sein muß. Die Adresse X ist dezimal zu verstehen 0 . . . 65535, keine Zeilennummer!); soll die Startadresse sedezimal genannt werden, ist dies mit HEX ("X") möglich. Nach dem Rücksprung aus dem Maschinen-Unterprogramm geht die Programmausführung in Basic bei dem Befehl weiter, der hinter der CALL-Anweisung steht.

CHR$ (X)
Das der Zahl X entsprechende Zeichen nach ASCII wird ausgegeben. Bei Angaben für X, die außerhalb 0. . . 255 (8-Bit-Wortlänge) liegen, wird mit der Fehlermeldung „?FC Fehler" (FUNCTION CALL ERROR) die Bereichsüberschreitung gemeldet.

PRINT CHR$(65) <CR> Anzeige: A

CLEAR
Setzt alle Variablen auf Null. Vorhandene Speicherplatzreservierungen werden nicht gelöscht.

CLEAR X
Setzt alle Variablen auf Null und reserviert X Bytes für die Variablen-Speicherung; X kann eine (positive) Variable sein, darf aber die Anzahl der freien Speicherstellen nicht überschreiten, sonst erfolgt die Fehlermeldung „?OM Fehler" (OUT OF MEMORY). Zu kleine Reservierung ergibt „?OS FEHLER" (OUT OF STRING).

CLEAR 200 <CR>
PRINT FRE(A$) <CR> Anzeige: 200

CLRS
Löschen des Bildschirms

CONT
Veranlaßt den Interpreter, ein zuvor per STOP-Befehl abgebrochenes Programm beim nächstfolgenden Befehl fortzusetzen. In der Zwischenzeit dürfen die Variablen oder Zwischenergebnisse inspiziert werden, allerdings sind vor dem CONT-Befehl keine Änderungen im Programm zulässig, sonst erfolgt Fehlermeldung „?CN FEHLER" (CONTINUE ERROR). Beim INPUT-Befehl bewirkt die <CR>-Taste (Return-Taste) einen Abbruch (Meldung „o. k."). Mit CONT wird der INPUT-Befehl wiederholt.

COS (X)
Bildet den Kosinus vom Argument X, das im Bogenmaß angegeben oder umgerechnet werden muß.

PRINT COS(60*3.1415/180) <CR> Anzeige: .5002
PRINT COS(60) <CR> Anzeige: −.95241

CSAVE
Ausgabe des gesamten gespeicherten Programms auf Magnetband-Kassette (ohne Angabe eines Programmnamens und ohne Steuerung des Laufwerks). Es empfiehlt sich, das Programm doppelt abzuspeichern.

CLOAD
Einlesen eines auf Magnetband-Kassette gespeicherten Programms (ohne Angabe eines Programm-Namens und ohne Steuerung des Laufwerks). Ein zuvor im Speicher vorhandenes Programm wird gelöscht.

DATA X,Y,Z ...
Definiert eine Datenzeile mit verschiedenen, durch Komma getrennten Zahlenwerten.

Data A$, B$, C$ ...
Definiert eine Datenzeile mit verschiedenen, durch Komma getrennten Texten.

DEF FN A(X) = Y
Definiert den Variablennamen A als eine neue Funktion A, die die Rechenvorschrift Y (mathematischer Ausdruck) zusammenfaßt und diese beim Aufruf auf Variable anwendet. (X) ist ein Dummy-Argument, für das jeder beliebig alphanumerische Wert eingesetzt werden kann.

Beispiel:
```
10 DEF FN Q1(X)=X*B+B
20 INPUT A,B:PRINT FN Q1(A)
```

DIM X(z)
Reserviert für die numerische Variable "X" z Feldelemente, beginnend bei 0 und endend bei (z–1).

DIM X(i,j,k)
Reserviert für die numerische Variable "X" eine mehrdimensionale Matrix mit i*j*k Feldelementen, die jeweils mit dem Index 0 beginnen und mit (i–1) bzw. (k–1) enden.

DIM A$(X)
Reserviert für die String-Variable "AS" X Feldelemente, beginnend bei 0 und endend bei (X–1).

DIM A$(i,j,k)
Reserviert für die String-Variable "A$" eine mehrdimensionale Matrix mit i*j*k Feldelementen, die jeweils mit dem Index 0 beginnen und enden bei (i–1) bzw. (j–1) bzw. (k–1).

DRAWTO X,Y
Zeichnet eine Gerade vom augenblicklichen Standpunkt des Cursors nach (x,y). Der Punkt (x,y) wird neuer Standpunkt. Der Bildschirm hat 512*256 Punkte. Die linke untere Bildecke besitzt die Koordinaten 0,0. Mit dem PAGE-Befehl wird die Schreibseite voreingestellt.

END
Zeigt dem Interpreter an, daß dieser die Programmausführung beenden und in die Basic-

Anweisungs-Ebene zurückspringen soll. Dieses Statement ist entbehrlich, wenn am Programmende keine weiteren Programmzeilen folgen (z. B. die von Unterprogrammen); DATA-Anweisungen können hinter einem Programm stehen, ohne daß davor ein END eingefügt werden muß.

EXP (X)
Bildet die X-te Potenz zur Basis e (= 2.7182); bei zu großem X erfolgt die Fehlermeldung "?OV Fehler" (OVERFLOW).

PRINT EXP(20) <CR> Anzeige: 4.8516E+08

FRE (X)
Ermittelt ab Adresse 8800 H die (dezimale) Anzahl freier Speicherplätze. Bei der SBC-2-Baugruppe mit zwei Speicherbausteinen (RAM) ergibt sich jeweils nach dem Kaltstart folgende Zahl:

PRINT FRE(0) <CR> Anzeige: 1784

FOR X = A TO Z STEP N
Definiert den Anfang einer Programmschleife, in der die Laufvariable X die Werte von A bis Z annehmen und bei jedem Durchlauf um die Schrittweite N erhöht werden soll; im Falle A = Z wird die Schleife einmal durchlaufen, und bei fehlender Angabe der Schrittweite N wird die Zahl 1 angenommen. Schleifen können geschachtelt werden.

GOSUB X
Springt in das bei Zeile X beginnende Unterprogramm, aus dem bei Erreichen des RETURN-Befehls (s. u.) automatisch der Rücksprung ins aufrufende Programm erfolgt; die Programmausführung geht mit der nächsten Basic-Zeile weiter, es muß in der Zeile GOSUB X der letzte Befehl in einer Zeile sein.

GOTO X
Setzt die Programmausführung bei Zeile X fort (unbedingter Sprungbefehl).

HEX ("X") oder **HEX (A$)**
Setzt die Sedezimalzahl ("Hexzahl") "X" in das dezimale Äquivalent um; mehr als vierstellige Angaben führen zu Fehlinterpretationen. Die Sedezimalzahl 0–7FFF ergibt 0 . . . 32767, 8000–FFFF ergibt −32768 . . . −1. Die sedezimalen Zeichen A . . . F müssen in Großbuchstaben eingegeben werden.

IF X=Y THEN (GOTO) Z
Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X = Y erfüllt ist; andernfalls geht es bei der nächstfolgenden Basic-Zeile weiter (bedingter Sprung).

IF X<>Y THEN (GOTO) Z
Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X<>Y (X ungleich Y) erfüllt ist; andernfalls geht es bei der nächstfolgenden Basic-Zeile weiter (bedingter Sprung).

IF X<Y THEN (GOTO) Z
Setzt die Programmausführung bei Zeile Z fort, wenn die Bedingung X<Y erfüllt ist, andernfalls geht es bei der nächstfolgenden Basic-Zeile weiter (bedingter Sprung).

INP (X)
Liest Daten von demjenigen Eingabe-Kanal ein, dem die dezimale Adresse X (0...255) zugeordnet ist. Die Angabe einer sedezimalen Adresse ist durch HEX ("X") möglich.

INPUT X
Dieser Befehl ist nicht im Direkt-Modus (ohne Zeilennummer) anwendbar. Die CR-Taste (Return-Taste) bricht die Programmausführung ab (Fortsetzung durch Eingabe von CONT). Der Befehl erwartet die Eingabe einer numerischen Variablen, die anschließend unter der Bezeichnung "X" geführt wird. Es ist darauf zu achten, daß Nachkomma-Stellen nicht durch ein Komma, sondern einen Punkt abgetrennt werden, weil das Komma zur Trennung zweier aufeinanderfolgender Eingaben dient (Fehlermeldung „zu viel"). Nicht-numerische Eingaben (z. B. Buchstaben oder Sonderzeichen) werden zurückgewiesen (Fehlermeldung „neue Eingabe").

INPUT X,Y
Erwartet die Eingabe zweier numerischer Variablen, die durch ein Komma voneinander getrennt werden müssen und anschließend unter der Bezeichnung "X" "Y" geführt werden. Die Reaktionen auf Falscheingaben erfolgen sinngemäß wie bei INPUT X.

INPUT "ABC"; X
Wie INPUT X, aber mit vorheriger Ausgabe des Textes "ABC" auf dem Bildschirm, gefolgt vom Fragezeichen. Dies ist eine recht elegante Eingabeform, weil der Computer regelrecht nach der zu einem Text (z. B. „Lastwiderstand=?") gehörenden Zahl „fragt".

INPUT A$
Erwartet die Eingabe einer Zeichenkette mit max. 79 Zeichen, die beliebige Zeichen enthalten darf, also auch Ziffern, jedoch kein Komma, weil das zur Trennung zweier aufeinanderfolgender Eingaben dient. Soll der String auch ein Komma enthalten, ist er in Anführungszeichen zu setzen.

INT (X)
Liefert bei einer Gleitkommazahl X den nächstkleineren, ganzzahligen Zahlenwert (Integer).

PRINT INT(123.55)	Anzeige: 123
PRINT INT(−0.5)	Anzeige: 1

LEFT$ (A$,n)
Spaltet vom String A$ die ersten n Zeichen ab. Wenn der String nicht n Zeichen lang ist, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n<= 0 ergeben die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR).

PRINT LEFT$("Meier",3) <CR>	Anzeige: MEI

LEN (A$)
ermittelt die Anzahl der im String A$ enthaltenen Zeichen.

PRINT LEN("Meier") <CR>	Anzeige: 5

LET X = ABC
Weist der numerischen Variablen X den rechts vom Gleichheitszeichen stehenden Wert zu. Der Begriff „LET" ist hierbei zwar entbehrlich, kann unter Umständen aber die Übersichtlichkeit eines Programms erhöhen.

LET A$ = "ABC"
Weist der String-Variablen A$ die in Anführungszeichen stehende Zeichenkette zu. Auch hier ist der Begriff "LET" entbehrlich, dient aber u. U. der Übersichtlichkeit eines Programmes.

LIST
Bewirkt die Ausgabe des gesamten gespeicherten Programms zeilenweise auf dem Bildschirm und kann mit der Taste <ESC> abgebrochen werden.

LIST X
Wie LIST, jedoch beginnend bei der Zeilennummer X. Um nur einen kleinen Programmausschnitt ausgeben zu lassen, gibt man "LIST X" und <CR> ein, hält die CTRL-Taste während der Return-Auslösung gedrückt und betätigt sofort (zusätzlich zur CTRL-Taste) zum Anhalten die Taste "S" und zum weiteren Anschauen die Taste "Q".

LLIST
Wie LIST, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LLIST X
Wie LIST X, nur erfolgt die Ausgabe auf dem Drucker.

LOG (X)
Bildet den natürlichen Logarithmus von X (zur Basis e = 2.7182); bei negativem X erfolgt die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR); bei zu großem X die Meldung "?OV Fehler" (OVERFLOW).

PRINT LOG(1000) <CR>　　　　　Anzeige: 6.9077

LPRINT X
Wirkt genauso wie PRINT, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.
Für LPRINT gilt die abkürzende Form des Fragezeichens nicht.
Das diesem Befehl vorangestellte "L" steht als Abkürzung für engl. „Lineprinter“ (Zeilendrukker).

LPRINT "XYZ"
Wie PRINT "XYZ", nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LPRINT A$
Wie PRINT A$, nur erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LPRINT X;Y
Im Gegensatz zu PRINT X;Y erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

LPRINT X,Y
Im Gegensatz zu PRINT X,Y erfolgt die Ausgabe nicht auf dem Bildschirm, sondern auf dem Drucker.

MID$ (A$,n,m)
Löst aus dem String A$, beginnend beim n-ten Zeichen, m Zeichen heraus. Wenn nicht n oder m Zeichen vorhanden sind, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n bzw. m <= 0 ergeben die Meldung "?FC Fehler" (FUNCTION CALL ERROR).

PRINT MID$("ABCDEFGH",2,3) <CR>   Anzeige: BCD

MOVETO X,Y
Nur bei Grafikausgabe wird der Bildpunkt mit den Koordianten X,Y neuer Cursor-Standpunkt, von dem aus mit dem Befehl DRAWTO U,V eine Linie zum Bildpunkt an der Stelle U,V gezeichnet wird.

NEW
Initalisiert den Interpreter neu (Buffer und Variablen löschen, internen Stack definieren u. a.) und wirkt wie das Löschen des Programmspeichers. Tatsächlich aber wird der Programmspeicher nicht gelöscht, sondern an den Beginn des Programmspeichers (die Startadresse steht in den RAM-Zellen 8857H/58H) werden zwei Bytes "00" eingeschrieben. Außerdem wird das Ende des Programmspeichers (die Endadresse steht in den RAM-Zellen 8887H/88H) hinter das zweite gelöschte Byte gesetzt (also um zwei Plätze höher als der Programmanfang). Das bedeutet, daß das Programm (bis auf die ersten beiden Bytes) noch im Speicher steht, vom Interpreter aber nicht mehr ausgeführt werden kann, weil Anfangs-und Endmarkierung durch die NEW-Anweisung zusammengelegt worden sind. Beim Einbau in ein Programm wirkt dieser Befehl wie ein Software-Selbstmord: Das Programm löscht sich quasi selbst aus.

NEXT X
Definiert das Ende der mit FOR . . . TO . . . STEP begonnenen Programmschleife. Die Nennung der Laufvariablen X kann hierbei entfallen.

NOT X
Invertiert im binären Äquivalent von X jedes einzelne Bit; durch die dezimale Anzeige der Ergebnisse kommt es oft zu Mißverständnissen, weil das Ergebnis von "NOT 55" gleich –56 ist. Das liegt daran, daß der Interpreter ein HIGH im Höchstwertigen Bit eines Ergebniswortes als negatives Vorzeichen wertet.

NULL X
Der Befehl NULL X <CR> verschiebt den linken Rand auf dem Bildschirm um X Zeichen nach rechts. Längere Zeichenketten werden auf der linken Bildsschirmseite fortgesetzt.

ON X GOTO A,B,C . . .
Setzt die Programmausführung bei Zeile A fort, wenn X = 1 ist, bei Zeile B, wenn X = 2 ist, bei Zeile C, wenn X = 3 ist, usw. (bedingte Verzweigung).

X OR Y
Bildet auf Maschinen-Ebene bitweise die ODER-Verknüpfung aus dem binären Äquivalent von X und Y, d. h. im Ergebniswort steht überall dort eine 1, wo an der korrespondierenden Stelle der Operanden X oder Y eine 1 war.

PRINT 37 OR 16 <CR>   Anzeige: 53

OUT X,Y
Gibt den zu Y gehörenden Zahlenwort (in binärer Form) an demjenigen Ausgabe-Kanal aus, dem die Adresse X zugeordnet ist; X ist hierbei dezimal zu verstehen (0 . . . 255; die Ausgabe einer sedezimalen Adresse ist durch HEX ("X") bzw. HEX ("Y") möglich).

PAGE N,M
Der GDP64 wird voreingestellt. Dabei wird auf die Seiten N geschrieben und die Seite M angezeigt. Nach diesem Befehl darf kein Input-Befehl folgen, da sonst die Seiteneinstellung nicht mehr stimmt. Mit dem Befehl PAGE 0,0 in einem Basic-Grafik-Programm wird der Bildaufbau sichtbar.

PEEK (X)
Liest den Inhalt der Speicherzelle mit der Adresse X; X ist hierbei dezimal zu verstehen (0 . . . 65535; die Angabe einer sedezimalen Adresse ist durch HEX ("X") möglich).

POKE X,Y
Schreibt den Wert Y in die Speicherzelle mit der Adresse X; X und Y sind dabei dezimal zu verstehen (0 . . . 65535 bzw. 0 . . . 255; die Angabe sedezimaler Zahlen ist durch HEX ("X") bzw. HEX ("Y") möglich).

POS (X)
Ermittelt die Cursor-Position in der laufenden Zeile, nennt also die Anzahl der bereits ausgegebenen Zeichen. (X) ist ein Dummy-Argument, für das jeder beliebige alphanumerische Wert eingesetzt werden kann.

PRINT X
Dient im Direkt-Modus dazu, nach Return unmittelbar eine Ergebnisanzeige auf dem Bildschirm zu erzeugen (ohne vorherigen Programmstart), z. B. das Ergebnis einer Rechenoperation oder die Darstellung von Zwischenergebnissen bzw. Variablen nach einer Programmunterbrechung. Dabei kann "X" eine im Programm verwendete Variable sein oder eine Rechenvorschrift (z. B. 275*1.14) oder bei "X$" eine Zeichenkette oder eine beliebige Zeichenfolge, die dann in Anführungszeichen zu setzen ist. Beim Einsatz dieses Befehls in einem Programm ergibt sich eine Fülle von Varianten. Zur Abkürzung kann man anstelle von "PRINT" einfach ein Fragezeichen "?" eingeben.

PRINT "XYZ"
Bewirkt die Ausgabe der in Anführungszeichen stehenden Zeichenkette auf dem Bildschirm.

PRINT A$
Bewirkt die Ausgabe der zur String-Variablen A$ gehörenden Zeichenkette auf dem Bildschirm.

PRINT X;Y
Bewirkt die Ausgabe der entsprechenden Zahlenwerte für X und Y hintereinander. Somit können auch numerische und String-Variablen zusammen ausgegeben werden, z. B. das Ergebnis einer Rechnung mit einem passenden Text. Zwischen beide Ausgaben wird ein Leerzeichen zur Trennung eingefügt; vor positiven Zahlenwerten steht ein weiteres Leerzeichen, weil das positive Vorzeichen unterdrückt wird.

PRINT X,Y
Bewirkt die Ausgabe der entsprechenden Zahlenwerte in einer Zeile, wobei die zweite (und jede folgende) Ausgabe bei der nächsten Tabulator-Position anfängt (neue Tabulator-Position: alle 14 Spalten).

READ N
Ruft das jeweils nächste Daten-Element (in diesem Fall eine Zahl) ab; bei jeder Ausführung des READ-Befehls wird ein interner Daten-Pointer um eins erhöht, um für den folgenden Zugriff das nächste Element zu adressieren. Findet der Interpreter kein Daten-Element mehr, weil mehr READ-Befehle ausgeführt wurden als Daten bereitgestellt sind, erfolgt die Fehlermeldung "?OD Fehler" (OUT OF DATA).

READ N$
Wie READ N, jedoch hier Abruf von Texten.

REM
Kleinbuchstaben bleiben nach dem REM-Befehl erhalten. Dieser definiert die laufende Zeile als Kommentarzeile, d. h. nach "REM" kann jeder beliebige, erläuternde Text stehen. Der Interpreter übergeht eine Kommentarzeile bei der Programmausführung, jedoch kann eine solche Zeile als Sprungziel dienen, sollte aber nicht, weil REM-Zeilen keine Befehle enthalten.
10 REM Gitternetz oder 10 PRINT A: REM Ausgabe.

RESTORE
Bewirkt das Rücksetzen des DATA-Zählers, der mit jedem READ-Befehl um Eins erhöht wird und damit stets auf die nächste, per DATA definierte Konstante weist. Wird bei READ keine durch DATA definierte Variable mehr gefunden, erfolgt die Fehlermeldung "?OD Fehler" (OUT OF DATA); das vorherige Rücksetzen per RESTORE vermeidet dies.

RETURN
Schließt ein Basic-Unterprogramm ab und bewirkt den Rücksprung an die zuvor mit dem Befehl GOSUB verlassene Stelle im aufrufenden Programm (s. o.).

RIGHT$ (A$,n)
Spaltet vom String A$ die letzten n Zeichen ab. Wenn der String nicht n Zeichen lang ist, werden entsprechend weniger genommen, ohne daß eine Fehlermeldung erfolgt. Zahlen für n <= 0 ergeben die Fehlermeldung "?FC Fehler" (FUNKTION CALL ERROR).

RND (X)
Erzeugt eine Gleitkomma-Pseudo-Zufallszahl zwischen 0...1; es ist sichergestellt, daß bei verschiedenen Programmdurchläufen nicht jedesmal dieselbe Zahl bzw. dieselbe Zahlenfolge auftritt. (X) ist ein Dummy-Argument. Negative Zahlen ergeben konstante Pseudozufallszahlen.

PRINT RND(−1) <CR> Anzeige: 7.6594 E−06

RUN
Veranlaßt den Interpreter, ein gespeichertes Programm auszuführen, beginnend bei der niedrigsten Zeilennummer.

RUN X
Veranlaßt den Interpreter, ein gespeichertes Programm auszuführen, beginnend bei der Zeilennummer X.

SGN (X)
Liefert das Vorzeichen von X; + 1 bei positivem X, − 1 bei negativem X und Null bei X = 0 (Signum-Funktion).

SIN (X)
Bildet den Sinus vom Argument X, das im Bogenmaß angegeben oder umgerechnet werden muß.

PRINT SIN(45*3.1415/180) <CR> Anzeige: .70709
PRINT SIN(45) <CR> Anzeige: .8509

SPC (X)
Rückt den Cursor um X Stellen nach rechts und füllt den Zwischenraum mit Leerzeichen (Blanks) auf.

SQR (X)
Bildet die Quadratwurzel aus dem (positiven) Argument X. Bei negativem X erfolgt die Fehlermeldung "?FC Fehler" (FUNCTION CALL ERROR) für die Bereichsüberschreitung und bei zu großem X wird "?OV Fehler" (OVERFLOW) angezeigt, die Bezeichnung "SQR" steht als Abkürzung für engl. "Square Root" = Quadratwurzel.

PRINT SQR(2) <CR> Anzeige: 1.4142

STR$ (X)
Wandelt die numerische Variable X in eine String-Variable um und ermöglicht damit die Anwendung von String-Operationen auf Zahlen. Nach erfolgter Umwandlung kann mit der neu definierten String-Variablen keine mathematische Operation mehr durchgeführt werden, auch wenn es sich augenscheinlich um eine Zahl handelt.

STOP
Bewirkt nach der Ausführung dieses Befehls die Unterbrechung des Programms mit der Meldung "abgebrochen in Zeile xyz"; anschließend ist die Inspektion und Modifikation von Variablen mit darauffolgender Fortsetzung des Programms möglich (per CONT), allerdings darf dabei das Programm selbst nicht modifiziert werden (programmierte Programmunterbrechung).

TAB (X)
Rückt den Cursor vom linken Bildrand aus um X Stellen nach rechts. Im Beispiel kann daher der zweite TAB-Befehl nicht ausgeführt werden.

PRINT TAB(10);"*";TAB(5);"*" <CR> Anzeige: **

TAN (X)
Bildet den Tangens vom im Bogenmaß angegebenen Wert X.

PRINT TAN(45*3.1415/180) <CR> Anzeige: .99995
PRINT TAN(45) <CR> Anzeige: 1.6197

USR (X)
Ruft die bei der Adresse X beginnende Anwender-Funktion auf und übergibt (im Gegensatz zu CALL) das Ergebnis im Gleitkomma-Akkumulator des Interpreters (Adressen 886E . . . 8870H für Mantisse und Vorzeichen und Adreese 8871 H für den vorzeichenbehafteten Exponenten).

VAL (A$)
Wandelt die String-Variable A$ in eine numerische Variable um und ermöglicht anschließend wieder die Anwendung mathematischer Operationen. Von A$ wird allerdings nur derjenige numerische Anteil berücksichtigt (Folge von Zahlen), der keine ASCII-Zeichen enthält, d. h. alle Zeichen, die im ursprünglichen String A$ rechts vom ersten nicht-numerischen Zeichen stehen, werden bei der Typ-Umwandlung nicht berücksichtigt.

PRINT VAL("1234ABCD") <CR> Anzeige: 1234
PRINT VAL("ABC12") <CR> Anzeige: 0

WAIT X,Y,Z
Wie INP X, aber mit anschließender Exklusiv-ODER-Verknüpfung mit Z, gefolgt von der UND-Verknüpfung mit Y. Der Interpreter führt den nächsten Befehl erst dann aus, wenn das so entstandene Ergebnis ungleich Null ist. Im Falle Z = 0 (oder bei fehlender Z-Angabe) werden die eingelesenen Daten nur mit Y UND-verknüpft. Die angegebenen Werte für X, Y und Z sind dezimal zu verstehen (0 . . . 255); die Angabe von sedezimalen Operanden ist durch HEX ("X") bzw. HEX ("Y") bzw. HEX ("Z") möglich.

Der Befehl WAIT 116,4,0 bewirkt eine Warteschleife, die erst verlassen wird, wenn Bit 2 des Statusregisters mit der Adresse 70H des Grafikprozessors EF9366 den Wert 1 besitzt.

Fehlererkennung und Fehlermeldung

Bei der Umsetzung und Ausführung eines gespeicherten Programms prüft der Interpreter ständig, ob bei der Formulierung der einzelnen Befehle das vorgeschriebene Eingabeformat eingehalten worden ist (z. B. Setzen mit Klammern, Anhängen eines Dollar-Zeichens o. ä.). Derartige Syntax-Fehler sind für das Programm ohne Schwierigkeiten erkennbar, weil es die Eingaben nur mit einem vorgegebenen Muster zu vergleichen braucht.

Darüber hinaus gibt es Fehler des Bedieners, die eine Programmausführung unmöglich machen (z. B. wenn der Programm- oder Variablenspeicher voll ist und keine Daten mehr aufgenommen werden können). Ebenso gehört die Prüfung auf verbotene Rechengänge (Teilen durch Null, Wurzel aus negativer Zahl ziehen), fehlende Definitionen oder Bereichsüberschreitungen zu den Überwachungsaufgaben des Interpreters. Fehlermeldung: ?FF FEHLER (IN ZEILE XYZ).

Es wurde einer der obengenannten Fehler erkannt, für dessen Identifikation ein zweistelliger Fehlercode "FF" ausgegeben wird.

Im Direkt-Modus erfolgt nur die Ausgabe "?FF Fehler", während im Programmlauf auch noch die Zeilennummer angegeben wird, in der der Fehler auftritt: "?FF Fehler in Zeile CYZ"; bis auf die (international üblichen) zweistelligen Fehlercodes erfolgt diese Meldung in deutsch. Nach einer derartigen Fehlermeldung wird ein Programmlauf sofort abgebrochen, und der Interpreter geht zurück in den Basic-Anweisungs-Modus (o.k.-Meldung).

Zuviel

Wenn bei der INPUT-Anweisung, die eine Bediener-Eingabe verlangt, mehr Eingaben gemacht werden als angefordert (mehrere Eingaben werden durch Komma voneinander getrennt), dann erfolgt die Meldung „zuviel". In diesem Fall ignoriert der Interpreter die überzähligen Eingaben und fährt mit der Programmausführung fort.

Diese Meldung ergeht auch dann, wenn eine Dezimalzahl anstelle des Dezimalpunktes ein Komma enthält, da der Interpreter dies als zwei Eingaben versteht (Trennzeichen Komma).

Neue Eingabe

Wenn bei Eingaben der falsche Variablen-Typ gewählt wird (bei erwarteten numerischen Daten die Eingabe von Buchstaben oder umgekehrt), dann erfolgt die Meldung „neue Eingabe". In diesem Fall ignoriert der Interpreter die falschen Eingaben und erwartet die Eingabe des richtigen Datentyps; danach fährt er mit der Programmausführung fort.

Liste und Bedeutung der Fehlermeldungen

BS BAD SUBSCRIPT. Undefiniertes Feldelement: falsche Indizierung; ein aufgerufenes Matrix-Element liegt außerhalb der durch DIM festgelegten Grenzen.

CN CONTINUE ERROR, kein CONT möglich: Die Fortsetzung eines zuvor unterbrochenen Programms per CONT ist nicht möglich, weil entweder ein Fehler vorliegt oder das Programm selbst zwischenzeitlich modifiziert worden ist.

DD DOUBLE DIMENSION, Mehrfachdefinition eines Feldes: Dasselbe Feld wird im Programm noch einmal dimensioniert.

FC FUNCTION CALL ERROR, Rechenfehler: Bei einem Funktionsaufruf liegt ein Parameter außerhalb des zulässigen Bereichs, z. B. beim Wurzelziehen aus einer negativen Zahl.

ID ILLEGAL DIRECT, als Direktbefehl nicht erlaubt: Die gewählte Anweisung ist im Direkt-Modus nicht zulässig; eine Funktions-Definition beispielsweise kann nur innerhalb eines Programms, nicht aber im Direkt-Modus aufgerufen werden.

LS LONG STRING, String zu lang: Ein String überschreitet die maximal zulässige Länge von 255 Zeichen.

NF NEXT WITHOUT FOR, Next ohne for: Eine Programmschleife ist unvollständig programmiert worden.

OD OUT OF DATA, zu wenig Daten: Es wurden mehr READ-Befehle ausgeführt als Daten bereitgestellt waren. Entweder müssen mehr Daten eingeführt werden oder der Daten-Pointer ist mittels RESTORE an den Beginn der Datenreihe zurückzusetzen.

OM OUT OF MEMORY, Variablenspeicher voll: Ein Bereich des Arbeitsspeichers ist voll; das kann ein zu langes Programm sein, eine Überschreitung des Variablen-Speichers oder auch eine Überfüllung des vom Interpreter benutzten Stacks (z. B. durch zu viele ineinander verschachtelte Unterprogramme oder FOR . . . NEXT-Schleifen).

OS OUT OF SPACE, Stringspeicher voll: Der für Strings reservierte Speicherbereich ist voll; das kann durch zu viele oder zu lange Strings passieren.

OV OVERFLOW, Überlauf beim Rechnen: Das Ergebnis einer mathematischen Operation überschreitet den max. möglichen Zahlenbereich.

RG RETURN WITHOUT GOSUB, Return ohne gosub: Es taucht ein RETURN-Befehl auf, ohne daß zuvor ein Unterprogramm-Aufruf erfolgt ist.

SN SYNTAX ERROR, Syntax-Fehler: Es liegt eine Verletzung des vorgeschriebenen Eingabe-Formats vor, z. B. die Eingabe "CHR(X)" statt "CHR$(X)".

ST STRING TOO COMPLEX, Fehler bei Stringverarbeitung: Ein String ist zu lang; er muß kürzer gefaßt oder in mehrere kürzere aufgeteilt werden.

TM TYPE MISMATCH, unterschiedliche Variablentypen: In einer Zuweisung kollidieren unterschiedliche Variablen-Typen; anstelle einer erwarteten numerischen Variablen wurde ein String übergeben oder umgekehrt.

UF UNDEFINED FUNCTION, undefinierte Funktion: Für eine im Programm aufgerufene Funktion fehlt zuvor die entsprechende Definition.

US UNDEFINED STATEMENT, Sprungziel fehlt: Es wurde eine falsche, nicht im Basic-Befehlssatz enthaltene Anweisung eingegeben (bzw. eine richtig gemeinte Anweisung wurde falsch geschrieben) oder es wurde im Programm ein Sprungziel aufgerufen, das gar nicht existiert.

/0 DIVISON BY ZERO, Teilung durch 0: Jemand hat verbotenerweise versucht, durch Null zu teilen.

Kleine Beispielprogramme für Grafik

Die nächsten beiden Seiten zeigen, wie man von Basic aus die GDP64 ansprechen kann.

Abb. 8.5.2

```
 1 REM GITTERNETZ 25*25 FELDER
 5 CLRS
10 PAGE 0,0                          :REM Schreibseite 0
20 FOR I = 0 TO 500 STEP 20          :REM senkrechte Linien
30 MOVETO I,0
40 DRAWTO I,250
50 NEXT
60 FOR I = 0 TO 500 STEP 10          :REM waagrechte Linien
70 MOVETO 0,I
80 DRAWTO 500,I
90 NEXT
100 GOTO 100
```

Abb. 8.5.3

zu Abb. 8.5.1

```
 5 CLRS:PAGE 0,0
10 X = RND(1)*511
20 Y = RND(1)*250
30 MOVETO X,Y
40 GOSUB 100
```

```
50 GOTO 10
100 REM WUERFEL
110 X = X + 10:GOSUB 200
120 Y = Y +  5:GOSUB 200
130 X = X - 10:GOSUB 200
140 Y = Y -  5:GOSUB 200
200 DRAWTO X,Y:RETURN
```

Abb. 8.5.4

```
10 CLRS:PI = 3.1415
20 INPUT"Anzahl der Schwingungen";S: S = S*2
30 PAGE 0,0:FOR X = 0 TO 511
40 Y = INT(SIN(X/511*PI*S)*120+120)
50 MOVETO X,120:DRAWTO X,Y
60 NEXT
70 POKE HEX("87C5"),0                 :REM Cursor aus
```

Abb. 8.5.5

```
10 CLRS: PAGE 0,0
20 FOR X = 0 TO 511
25 PI = 3.1415
30 Y = INT(SIN(X/511*PI*4)*COS(X/511*17*PI)*100+100)
40 Y1 = INT(SIN(X/511*PI*4)*100+100)
50 MOVETO X,100                       :REM Startpunkt
60 DRAWTO X,Y                         :REM Zielpunkt
70 MOVETO X,Y1:DRAWTO X,Y1
75 NEXT X
80 POKE HEX("87C5"),0
```

Abb. 8.5.6

```
10 CLEAR 100: CLRS
20 GDP=HEX("70")
30 INPUT "TEXT:";A$
40 INPUT "koordinaten X,Y:";X,Y
50 PAGE 0,0
60 OUT GDP+3,7+7*16:             REM Port 73 = Vergrößerung
70 MOVETO X,Y
80 FOR I = 1 TO LEN(A$)
85 OUT GDP,ASC(MID$(A$,I,1)):    REM Ausgabe der ASCII-Werte PORT 70
90 WAIT GDP,4,0:                 REM Warteschleife bis GDP fertig
100 NEXT I
110 OUT GDP+3,1+16               REM alte Schriftgröße
120 POKE HEX("87C5"),0:          REM Cursor aus
130 INPUT"Nochmal";X$
140 IF X$ = "J" THEN 10
```

Abb. 8.5.7

```
10 REM Hex-Monitor
20 CLEAR 100:DIM H$(16):DATA 0,1,2,3.4,5,6,7,8,9,A,B,C,D,E,F
30 FOR I = 0 TO 15:READ H$(I):NEXT I
```

zu Abb. 8.5.1

```
50 TZ$="0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F"
60 TS$="dez.  hex.   ":CLRS
100 PRINT"Speicher anschauen   =  1":PRINT
110 PRINT"Speicher aendern     =  2":PRINT
130 INPUT"Zahl";A: IF A>2 OR A<1 THEN 60
1000 CLRS: INPUT"ab Hex-Adr";B$:B = HEX(B$):PRINT
1015 IF B<0 THEN B=65536+B
1016 IF A=2 THEN 2000
1018 B=INT(B/16)*16
1020 INPUT"bis:";C$:C = HEX(C$):CLRS
1025 IF C<0 THEN C = 65536 + C
1030 PRINT TS$;TZ$
1040 FOR I = B TO C
1080 IF I/16 = INT(I/16) THEN PRINT:ZR=I:GOSUB 6010
1085 D=PEEK(I):S=0:GOSUB 5017
1090 NEXT I:PRINT:INPUT"nochmal";X$
1100 IF LEFT$(X$,1) ="j" THEN 1000
1110 GOTO 60
1500 REM AENDERN
2000 ZR=B:GOSUB 6010
2003 D=PEEK(B):S=0:GOSUB 5017
2005 INPUT"hex";F$:F=HEX(F$):IF F>255 THEN 2005
2010 POKEB,F:B=B+1:GOTO 2000
3000 REM DEZ-HEX
5000 S = INT(D/256):gosub 5100
5015 PRINT H$;
5017 S = D - S*256:GOSUB 5100
5020 PRINT H$ + "  ";:RETURN
5100 L = S AND 15: H = (S AND 240)/16
5150 H$ = H$(H)+H$(L):RETURN
6000 REM Format
6010 ZR$=STR$(ZR ):PRINT MID$(ZR$,2,6);SPC(8-LEN(ZR$));
6020 D=ZR:GOSUB 5000
6030 RETURN
```

Abb. 8.5.1 Einige Beispiele, wie man in Basic mit der Grafik umgeht (Programmlisting)

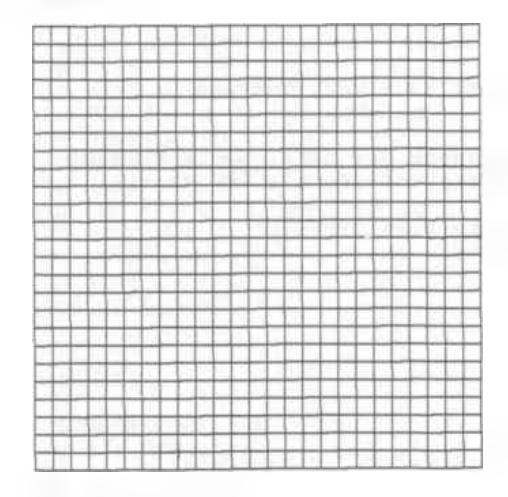


Abb. 8.5.2 Das Gitternetz

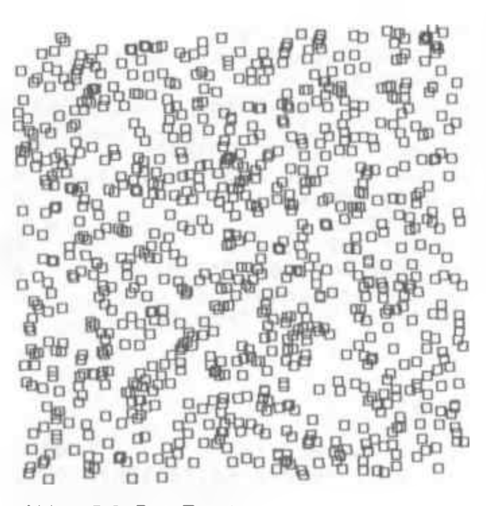


Abb. 8.5.3 Das Ergebnis eines Laufes von Zufallsquadraten

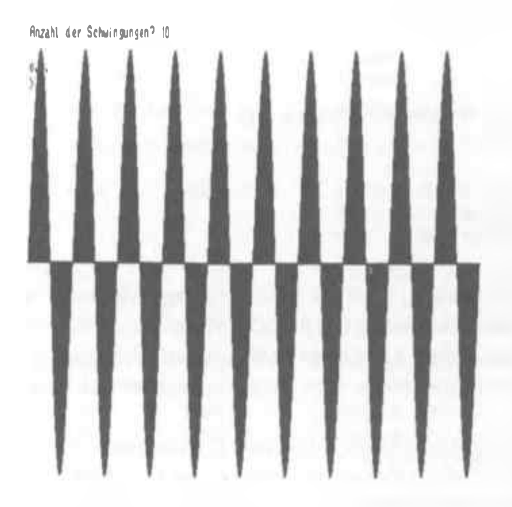


Abb. 8.5.4 Sinusschwingungen

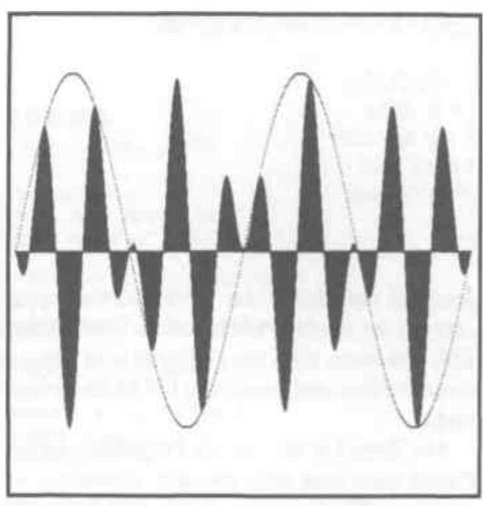
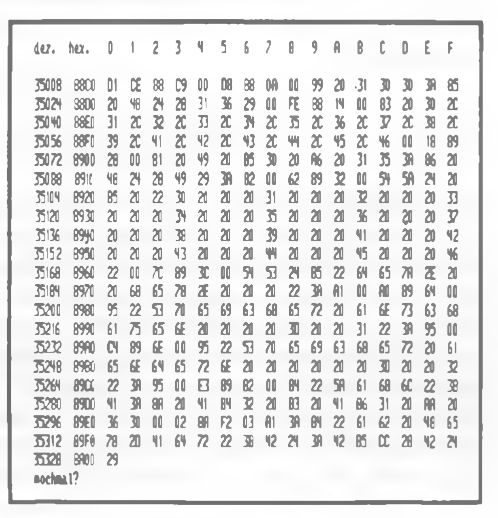


Abb. 8.5.5 Sinusgrafik



Abb. 8.5.6 Texte können vergrößert werden



dez.	hex.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
35008	88C0	D1	CE	88	C9	00	D8	88	0A	00	99	20	·31	30	30	3A	85
35024	88D0	20	48	24	28	31	36	29	00	FE	88	14	00	83	20	30	2C
35040	88E0	31	2C	32	2C	33	2C	34	2C	35	2C	36	2C	37	2C	38	2C
35056	88F0	39	2C	41	2C	42	2C	43	2C	44	2C	45	2C	46	00	18	89
35072	8900	28	00	81	20	49	20	B5	30	20	A6	20	31	35	3A	86	20
35088	8910	48	24	28	49	29	3A	B2	00	62	89	32	00	54	5A	24	20
35104	8920	B5	20	22	30	20	20	20	31	20	20	20	32	20	20	20	33
35120	8930	20	20	20	34	20	20	20	35	20	20	20	36	20	20	20	37
35136	8940	20	20	20	38	20	20	20	39	20	20	20	41	20	20	20	42
35152	8950	20	20	20	43	20	20	20	44	20	20	20	45	20	20	20	46
35168	8960	22	00	7C	89	3C	00	54	53	24	B5	22	64	65	7A	2E	20
35184	8970	20	68	65	78	2E	20	20	20	22	3A	A1	00	A0	89	64	00
35200	8980	95	22	53	70	65	69	63	68	65	72	20	61	6E	73	63	68
35216	8990	61	75	65	6E	20	20	20	20	30	20	20	31	22	3A	95	00
35232	89A0	C4	89	6E	00	95	22	53	70	65	69	63	68	65	72	20	61
35248	89B0	65	6E	64	65	72	6E	20	20	20	20	20	20	30	20	20	32
35264	89C0	22	3A	95	00	E3	89	B2	00	B4	22	5A	61	68	6C	22	3B
35280	89D0	41	3A	8A	20	41	B4	32	20	B3	20	41	B6	31	20	AA	20
35296	89E0	36	30	00	02	8A	F2	03	A1	3A	B4	22	61	62	20	48	65
35312	89F0	78	20	41	64	72	22	3B	42	24	3A	42	B5	CC	28	42	24
35328	8A00	29															

nochmal?

Abb. 8.5.7 Die Ausgabe des Hexmonitors

8.6 Flomon, das Z80-Monitorprogramm für die Floppy

Flomon sitzt in einem 8-KByte-EPROM auf der Bank/Boot-Baugruppe. Flomon beinhaltet Unterprogramme für die Floppy, wie Lesen und Schreiben, sowie Unterprogramme für die Bildschirmverwaltung. Dabei sind aber auch komfortable Grafik-Befehle eingebaut, die mit Steuerzeichen erreichbar sind. Dadurch ist es möglich, auch von beliebigen Programmiersprachen aus auf die leistungsfähige Grafik der GDP-Baugruppe zuzugreifen.

Wenn man den Z80-Computer mit Flomon startet, so meldet er sich wie in *Abb. 8.6.1* gezeigt. Man hat dann vier Auswahlmöglichkeiten. Wenn man die Taste 1 drückt, so wird die Floppy

```
FLOMON 4.0 Rolf-Dieter Klein (C) 1986

1 = Floppy Boot
2 = Go E0000h
3 = Go Bank 2000h
4 = ZEAT start
CTRL-C=Testmode

_
```

Abb. 8.6.1 So meldet sich Flomon 4.0

gestartet und davon das CP/M-Betriebssystem geladen. Welches Laufwerk angeschlossen ist, erkennt der Rechner automatisch. Damit kann man Laufwerke mit 8 Zoll, 5¼ Zoll, 3½ Zoll und 3 Zoll betreiben, alles mit einfacher oder doppelter Dichte. Allerdings muß dazu auf der Floppy das dazugehörige und angepaßte CP/M-Betriebssystem vorhanden sein. Dazu im nächsten Abschnitt mehr.

Mit Taste 2 kann man das Programm auf der Adresse E0000h, also Bank E, Adresse 0, starten. Damit kann man sich dort eine Sammlung von häufig gebrauchten Programmen (in EPROMs) anlegen. Mit der Taste 3 läßt sich der zweite EPROM-Platz auf der Bank-/Boot-Baugruppe starten, dort kann man z. B. das Grundprogramm (Version auf Adresse 2000h unterbringen. Mit der Taste 4 läßt sich das Christiani-Betriebssystem ZEAT starten. Mit den Tasten CTRL und C gelangt man in den Testmodus und kann die Befehle des Flomon prüfen, die nun besprochen werden sollen.

ASCII ein

Dazu drücken Sie bitte die Taste CTRL und dann die Taste „C". Es passiert zunächst nichts weiter, der Cursor blinkt noch. Wenn man nun aber einen Buchstaben drückt, zum Beispiel „A", so erscheint dieser auf dem Bildschirm. Sie befinden sich jetzt im sogenannten Alpha-Modus, indem die von der Tastatur kommenden Byte als ASCII-Zeichen interpretiert werden. Die Werte von 20h bis 7Fh sind druckende ASCII-Zeichen, die auf dem Bildschirm erscheinen. Die Codes 0 bis 1Fh sind nicht druckende Steuerzeichen. Davon dürften zumindest CR und LF schon bekannt sein. Sie haben den Wert 0Dh und 0Ah. Wenn man CR drückt, so wird der Cursor an die linke Seite des Bildschirms gerückt. Bei LF wandert eine Zeile nach unten. Es gibt beim Flomon viele solcher Zeichen. *Abb. 8.6.2* zeigt die Auflistung aller vorhandener Befehle. Die Zeichen 8, 9, 0Ah, 0Bh, 0Ch, 0Dh, 16h, 1Ah, 1Eh werden direkt interpretiert.

```
BEFEHLSBESCHREIBUNG  (C) 1984 Rolf-Dieter Klein
------------------------------------------------
 NDR-Klein-Computer FLOMON 1.5

Befehle
=======

ALPHA-MODUS
-----------

Ist nach dem Einschalten aktiv
20h..7fh                 Codes der sichtbaren ASCII-Zeichen

08h      CTRL H          Backspace
09h      CTRL I          Cursor Right
0ah      CTRL J          Linefeed
0bh      CTRL K          Cursor Up
0ch      CTRL L          Cursor Right
```

zu Abb. 8.6.2

```
0dh       CTRL M           Carriage Return
16h       CTRL V           Cursor Down
1ah       CTRL Z           Clear
1eh       CTRL ^           Home
Escape Sequenzen

1bh 3dh y x   ESC = r c    Cursor setzen (y,x =20h..7fh)

1bh 44h n     ESC D n      Lokal Mode einschalten.
                           ESC D L, schaltet den Mode ein
                           ESC D N, schaltet wieder aus.

1bh 51h       ESC Q        Zeichen bei Cursorposition einfügen

1bh 57h       ESC W        Zeichen bei Cursorposition löschen

1bh 45h       ESC E        Zeile bei Cursorposition einfügen

1bh 52h       ESC R        Zeile bei Cursorposition löschen
1bh 54h       ESC T        Zeile ab Cursorposition
                           bis zum Ende löschen

1bh 74h       ESC t        Zeile ab Cursorposition
                           bis zum Ende löschen

1bh 59h       ESC Y        Seite ab Cursorposition
                           bis Ende löschen

1bh 79h       ESC y        Seite ab Cursorposition
                           bis Ende löschen

1bh 7ah x     ESC z n      n='0' amerikanischer Zeichensatz;
                           n='1' deutscher Zeichensatz

Doppelescape Sequenzen

1bh 1bh 47h     ESC ESC G        Grafik-Modus

GRAFIK-MODUS
------------
Das Zeichen n steht für die Eingabe eines numerischen
Wertes. n kann eine vorzeichenbehaftete Dezimalzahl sein
oder eine sedezimale Zahl: 0, -50, $FF, -$4034.
Mehrere numerischen Werte werden durch Leerzeichen
getrennt.          Ferner sind
%dx %dy            relative Koordinaten mit Prozentzeichen
                   bei den Befehlen D,M,L,l moeglich.

CR ist das Abschlußzeichen.  Es kann bis auf
wenige Ausnahmen auch das Zeichen Strichpunkt (';')
verwendet werden.

A                  in den Alpha-Modus zurückschalten,

M n1 n2 CR         Positionieren auf x=n1 y=n2

D n1 n2 CR         Vektor nach x=n1, y=n2 zeichnen

m b1 b2 b3 b4      Wie M, jedoch mit binärer übertragung
                   und daher schnell.
                   x = b1,b2 (b1 ist höherwertiges Byte,
                              b2 ist niederwertiges Byte)
                   y = b3,b4

d b1 b2 b3 b4      Wie D, jedoch binär. Format wie
                   bei m
```

zu Abb. 8.6.2

J n1 n2 CR

Wie D, jedoch mit Angabe von
relativen Koordinaten
dx = n1 dy = n2

P n CR

Seite asynchron anwählen.
n = Nummer der Schreibseite * 4
+ Nummer der zu lesenden Seite
z.b. Schreibseite =3 , Leseseite = 1
n = 3*4 + 1 = 13. Befehl:
P 13 CR

S n CR

Seite synchron anwählen.
n wird wird wie oben bestimmt, die Seite wird
jedoch synchron zum Bildwechsel umgeschaltet.

X n CR

Die Seiten 0,1,2,3 werden
zyklisch angezeigt, n gibt die
Sichtdauer einer Seite an.
Die Anzeigedauer beträgt
n * 20ms
Wenn man den Befehl gegeben hat, so wird
auch der Alpha-Mode umgeschaltet, wenn
man mit A zurückkehrt, bis entweder ein
Bildschirmlöschen im Alpha-Mode stattfindet
oder X und Y im Graphikmode ohne Parameter
aufgerufen wurden, also z.B. X cr.
n = 0 beendet den Wechsel

Y n CR

Es werden jeweils nur zwei Seiten
zyklisch angezeigt und zwar die
Seiten 0 und 1, wenn eine dieser
beiden Seiten als Leseseite definiert
war, sonst 2 und 3, für den Alphamode gilt
das Gleiche, wie bei X. Wenn man z.B. X0 und
CR eingibt, so wird das automatische Seiten-
umschalten ausgeschaltet, auch im Alpha-Mode
und man Graphik und Text einfach mischen.
Der Cursor bleibt dann allerdings als heller
Block und blinkt nicht mehr.

C

Löschen der aktuellen Schreibseite

Z

Löschen aller Seiten 0 bis 3.
Die zuletzt angewähle Bildschirmseite
wird danach eingestellt.

G n1 n2 CR

Befehl an GDP
n1 = Nummer des GDP-Ports (0..15),
n2 = Datenwert an diesen Port

B text CR

Text (20h..7fh) an GDP-Port 0 senden.
Der Text wird wird ab aktueller
Koordinate des GDP ausgegeben

V binär 00h

Binärdaten an den GDP-Port 0 senden.
Das ASCII-Zeichen NUL beendet die
übertragung

O n1 n2 n3 n4 CR

Ellipsenabschnitte zeichnen.
Mit n1 wird die Länge der Halbachse
in x-Richtung angegeben, mit
n2 die Länge der Halbachse in
y-Richtung.

zu Abb. 8.6.2

```
                 Mit n3 wird der Startwinkel
                 bezüglich der x-Achse in Grad
                 angegeben. Mit n4 der Endwinkel
                 des Ellipsenabschnittes.
                 Der Ellipsenabschnitt wird von
                 der aktuellen x,y-Koordinate bis
                 zum Erreichen des Endwinkels
                 gezeichnet. Der Ellipsenmittelpunkt
                 wird aus der Startwinkel- und Halbachsen-
                 angabe vor Beginn des Zeichnens
                 automatisch errechnet.

O n1 n2 n3 n4 1 CR
                 Wie oben oben, jedoch der vom
                 Kurvenstück und den Radien zum
                 Mittelpunkt begrenzte Raum
                 gefüllt (Torte)

R n1 n2 CR       Rechteck ab aktueller x,y-Koordinate.
                 n1=dx und n2=dy geben die Breite und
                 die Höhe des Rechtecks an

R dx dy 1 CR     Rechteck gefuellt zeichnen. Sonst wie oben.

L n1 n2 n3 n4 ... nn nm CR
                 Polygon zeichnen, mit absoluten
                 Koordinaten.
                 x0=n1 y0=n2 gibt die Startposition an,
                 alle weiteren Paare geben die
                 Eckpunkte des Polygons an.
                 Der letzte Eckpunkt wird wieder
                 mit dem Startpunkt verbunden

l n1 n2 n3 n4 n5 n6 CR
                 Dreieck gefüllt zeichnen
                 x0 = n1  y0 = n2
                 x1 = n3  y1 = n4
                 x2 = n5  y2 = n6

F n1 n2 n3 CR    Fadenkreuz zeichnen,
                 an Position x=n1 y=n2,
                 auf Seite n3 (0..3).
                 Altes Fadenkreuz wird gelöscht.
                 Die Schreib- und Leseseite
                 bleiben erhalten.

F n1 n2 n3 rmw   CR
                 Fadenkreuz setzen, mit RMW
                 rmw=0 kein RMW-Zyklus
                 rmw=1 mit XOR-Mode
                 Nur mit Zusatzhardware auf der GDP
                 verwendbar.

r rmw CR         rmw=0 normal Mode
                 rmw=1 mit XOR-Mode arbeiten,
                 alle Befehle sind betroffen. Achtung, nur
                 mit Zusatzhardware. Bit 0 am Port 60h
                 bestimmt den RWM-Mode.
```

zu Abb. 8.6.2

```
r CR             rmw=0 setzen

T                Koordinate auf Stack. Die aktuellen Koordinaten
                 des GDP X und Y werden auf einen Stack gelegt.
```

	Bei zuvielen Koordinaten kommt es zum überlauf. Die Größe ist vom Restspeicher abhängig.
U	Koordinate von Stack. Die zuletzt mit T abgespeicherten Koordinaten werden zurückgeladen und die GDP-Position mit MOVETO eingestellt. Sind keine Koordinaten im Stack, so bleibt die Position unverändert.
Q	Stack loeschen. Der Koordinatenstack wird gelöscht. Dies ist zu Programmbeginn nützlich, um zu vermeiden, daß sich Koordinaten auf dem Stack ansammeln.
WA string CR	Symbol für den Fadenkreuz-Befehl umdefinieren. string ist eine Zeichenkette mit Zeichen im ASCII-Bereich (30h,31h,40h..5fh). 0 (30h) = Schreibstift hoch, 1 (31h) = Schreibstift runter. Der Code für die Schreibstiftbewegung berechnet sich wie folgt: Richtung + (8 * Länge) + 40h Es stehen die Richtungen von 0 bis 7 (* 45 Grad) zur Verfügung
WA CR	Symbol für den Fadenkreuzbefehl auf das Symbol "Fadenkreuz" zurücksetzen
WB	Fadenkreuzsymbol an aktueller x,y-Koordinate auf der aktuellen Schreibseite setzen
WC n1 n2 CR	Fadenkreuzsymbol vergrößern und drehen. n1 = Vergrößerungsfaktor (1..255) n2 = Drehung (0..7)
WC CR	Rückstellen auf n1 = 1, n2 = 0
WD n1 n2 n3 ... nn CR	Download. Es können die Daten n2 bis nn ab Adresse n1 in den Arbeitssppeicher geladen werden. Dabei kann man auf das normalerweise verborgene RAM auf der BANK/BOOT-Karte zugreifen.
WE n1 CR	Programm auf Adresse = n1 starten
H y0 CR	Füllen entlang der horizontalen Achse. Alle Vektorbefehle sind betroffen. Von da an wird die Fläche unter Vektoren bis zur angegebenen y-Achse ausgefüllt. (Befehl D,J und O sind betroffen).
H CR	Füllen ausschalten
I x0 CR	Füllen entalng der vertikalen Achse, wie H, jedoch mit X-Achse.
I CR	Füllen ausschalten

zu Abb. 8.6.2

```
Es folgen noch einige Befehle für die Farberweiterung. Dabei wird
über einen zusätzlichen Farbport (adresse 0A0h) der Farbcode
ausgegeben. Die Befehle wurden schon eingebaut, um möglichst
frühzeitig Kompatibilität zu erreichen.


WG farbe CR         Farbcode setzen
                    farbe: 0..255

7       6       5       4       3       2       1       0

DI weiß DW weiß DI blau DW blau DI grün DW grün DI rot  DW rot

0= aktiv geschaltet.
mit DW wird der Schreibvorgang eingeschaltet (=0)
mit DI kann Löschen je Farb-Ebene eingestellt werden (=1)

wichtige Codes fuer dominantes Schreiben:
Hintergrund:        Code    3F
weiß:               Code    C0
std:                Code    00
blaugruen:          Code    C3
violett:            Code    CC
blau:               Code    CF
gelb:               Code    F0
gruen:              Code    F3
rot:                Code    FC

K seite-A Farbe-A pen-A seite-B Farbe-B pen-B

                    Mehrseitenschreiben
                    Bereich seite: 0..3
                    Farbe = Farbcode
                    pen =0 bei schreiben =1 bei loeschen

K CR                stellt Normalzustand her

alle Linienzeichenbefehle und Zeichenausgabe sind betroffen


Monitoreinsprünge bei FLOMON -- Graphik-Routinen

F040:   JP CLRALL       ; alle Seiten loeschen
F043:   JP CLRINVIS     ; aktuelle WRT-Seite loeschen
F046:   JP MOVETO       ; hl=x de=y
F049:   JP DRAWTO       ; hl=x de=y
F04C:   JP WRTPG        ; c=Schreibseite
F04F:   JP VIEWPG       ; c=Leseseite
F052:   JP RMWPG        ; c=RMW-Mode (0,1)
F055:   JP WAIT         ; warten bis GDP fertig
F058:   JP CMD          ; c=Befehl fuer GDP
```

Abb. 8.6.2 Die Befehle von Flomon

Dann gibt es sogenannte Escape-Sequenzen. Dabei wird zunächst das Zeichen Escape (Taste mit der Beschriftung ESC oder CTRL und „eckige Klammer auf") gedrückt und dann ein weiteres Zeichen ESC und R (Achtung großes R, immer wie in *Abb. 8.6.2* angegeben) löscht zum Beispiel die Zeile, in der der Cursor steht. Ich empfehle jedem zunächst einmal alle Kombinationen einfach auszuprobieren, um ein Gefühl für den Befehlssatz zu gewinnen.

Da Flomon inzwischen eine neue Versionsnummer hat, zeigt *Abb. 8.6.3* die Unterschiede zur alten Version und *Abb. 8.6.4* die Schalter-Belegung der Key-DIL-Schalter.

1. Die Befehle WD, WE und WF wurden entfernt.
2. mit ESC) schaltet man die Unterstreichung ein
   mit ESC (wird die Unterstreichung ausgeschaltet.
3. Die Steprate kann nun minimal 3ms auch bei
   Minilaufwerken betragen.
4. Neue Scrollroutinen, es wird nur noch eine
   Bildseite für Textausgaben verwendet,
   um dadurch die Geschwindigkeit zu erhöhen.
   Gescrollt wird jetzt immer um 8 Zeilen.
5. Mit einem DIL-Schalter auf der KEY-Baugruppe
   können nun verschiedene Voreinstellungen
   durchgeführt werden (siehe Tabelle)
6. Die CAS-Baugruppe wird nicht mehr unterstützt,
   dafür ist nun die SER-Baugruppe als RDR: und
   PUN: Einsprung verfügbar.
7. Neuer Einsprung JP RISTS auf Adresse F05Eh,
   um Status der seriellen Schnittstelle prüfen zu
   können.
8. Befehl "ZEAT start" um Christiani Betriebsystem
   direkt starten zu können wurde eingebaut.

Abb. 8.6.3 Änderungen der Flomon-Version gegenüber 1.5

Steuern mit ESC

Mit „ESC = !!“ zum Beispiel kann man den Cursor in die erste Zeile und dort in die erste Spalte positionieren.

Die Steuersequenzen wurden nicht willkürlich gewählt, sondern entstammen einer in den USA verbreiteten Terminalgeneration der Firma Televideo: TVI 912, 920 und 950. Man kann so CP/M-Standard-Programme wie Wordstar verwenden, die diese Terminals meist direkt unterstützen.

Neben dem reinen Alpha-Modus gibt es aber auch noch einen speziellen Grafik-Modus. Dorthin gelangt man, wenn man zweimal ESC drückt und dann Shift und G, also großes G tippt. Mit Sift A, oder ESC ESC Shift A kommt man wieder in den alphanumerischen Betrieb zurück. Es gibt eine Vielzahl von Grafik-Befehlen. Wenn man in den Grafik-Modus kommt, verschwindet zunächst der Cursor. Der alte Text aus dem Alpha-Modus bleibt jedoch erhalten. Nun kann man neue Befehle geben, z. B. folgende Sequenz:

```
M0 0       carriage return
D300 150   carriage return
```

carriage return ist die Taste CR. Wenn man diese Zeichen eingibt, ändert sich auf dem Bildschirm zunächst nichts. Erst nach der letzten Eingabe erscheint eine schräge Linie auf dem Bildschirm. Jetzt versuche man einmal folgendes:

```
ZP0
M300 150           carriage return    I10 0 100 0 0 100   carriage return
R100 50 1          carriage return    M20 200             carriage return
O-100 50 0 270 1   carriage return    G3 $44              carriage return
F200 20 0          carriage return    BHallo              carriage return
```

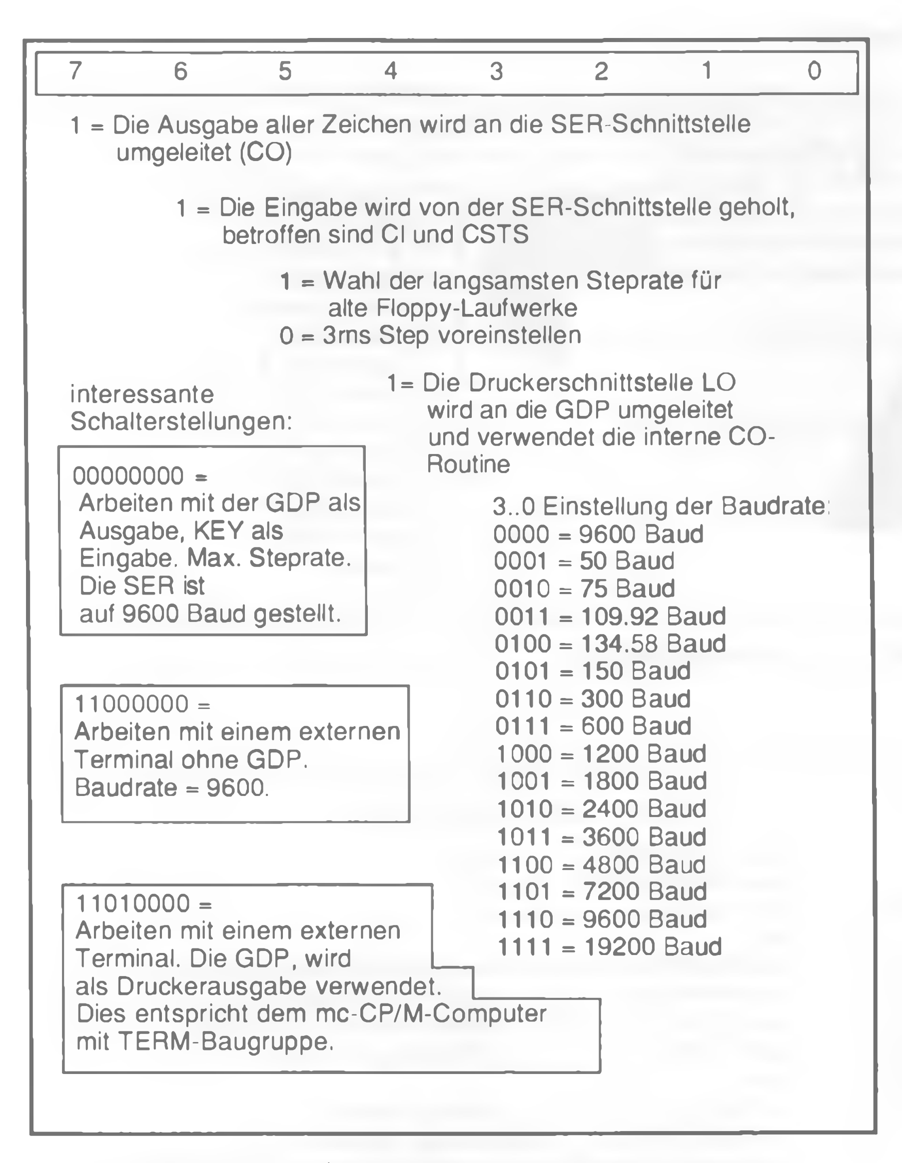


Abb. 8.6.4 Einstellung des DIL-Schalters auf der KEY-Baugruppe

Abb. 8.6.5 zeigt das Ergebnis, wenn Sie alles richtig getippt haben. Probieren Sie auch die anderen Befehle aus *(Abb. 8.6.2)*. All diese Befehle kann man, wie schon versprochen, von verschiedenen Sprachen aus verwenden, denn sie sind ja nur einfache Ausgaben an den Grafikprozessor. So zeigt *Abb. 8.6.6* ein Beispiel in Turbo-Pascal. Der Befehl Y0 bewirkt, daß das Grafik-Bild nicht blinkt, denn in Alpha-Modus werden normalerweise zwei Bildseiten verwendet, die im Cursortakt umgeschaltet werden. Mit Y0 kann man das abstellen. Dann blinkt allerdings auch der Cursor nicht mehr. Wenn man nur Y und CR eingibt, erhält man wieder den üblichen Umschaltwert. Mit Y1 ergibt sich eine ganz hohe Umschaltfrequenz. Die Befehle Y und X kann man für einen automatischen Bildwechsel verwenden (was z. B. für Phasenzeichnungen interessant ist).



Abb. 8.6.5 Eine Grafik-Demonstration

```
     Line 1    Col 1    Insert    Indent  A:TEST.PAS
|

program flomon(output);
begin
 writeln(chr(27),chr(27),'G');
 writeln('ZP0');
 writeln('M300 150');
 writeln('R100 50 1');
 writeln('O-100 50 0 270 1');
 writeln('F200 20 0');
 writeln('L0 0 100 0 0 100');
 writeln('M20 200');
 writeln('G3 $44');
 writeln('BHallo');
 writeln('Y0');
 writeln('A');
end.
```

Abb. 8.6.6 Ein Testprogramm in Turbo-Pascal

```
     Line 20   Col 6   Insert    Indent  A:TEST.PAS
program flomon(output);
var x,y:integer; ch:char;
 key : text;
begin
 assign(key,'KBD:'); reset(key); { fuer Einzelzeichen ohen Bildschirmausgabe}
 writeln(chr(27),chr(27),'G');   { Graphik-Mode einschalten }
 writeln('ZP0;Y1;M0 0;R20 10');  { Wechselfrequenz und kleines Rechteck }
 x:=256; y:=128;                 { Startposition des Fadenkreuzes }
 repeat
  writeln('F',x,' ',y,' 1');     { Fadenkreuz ausgeben }
  writeln('G0 128');             { dort immer einen Punkt setzen }
  read(key,ch); case ch of       { ein zeichen einlesen, ohne Bildschirmausgabe
   '+': y:=y+5; '-': y:=y-5;     { +- und <> steuern das Fadenkreuz }
   '<': x:=x-10; '>': x:=x+10;
  end;
 until (x<20) and (y<10);        { Stop , wenn im kleinen Rechteck angekommen }
 writeln('A');                   { zurueck in Alpha-Mode, nicht vergessen ! }
end.
```

Abb. 8.6.7 Der Fadenkreuz-Befehl in Turbo-Pascal

Es gibt übrigens auch ein Fadenkreuz (der Befehl F), das normalerweise auf Seite 1 ausgegeben wird und dann mit Y1 dem normalen Bild überlagert werden kann. Damit kann man einfache interaktive Eingabeprogramme schreiben. *Abb. 8.6.7* zeigt ein Programmbeispiel. Und *Abb. 8.6.8* eine mögliche Ausgabe davon. Damit das CP/M, genauer gesagt, das BIOS auf Flomon zugreifen kann, gibt es ein paar Einsprungstellen, die *Abb. 8.6.9* zeigt. Diese Einsprünge beinhalten Consol-Ein- und Ausgabe, so wie auch die Floppy-Routinen. Ferner gibt es auch für die Grafik noch direkte Einsprünge, wenn man besonders schnellen Zugriff auf die Grafik braucht (doch das nur für Spezialisten).

Der Einsprung FLOPPY ist wohl der wichtigste; wie er funktioniert, erfahren Sie im Abschnitt über das BIOS und das CP/M.

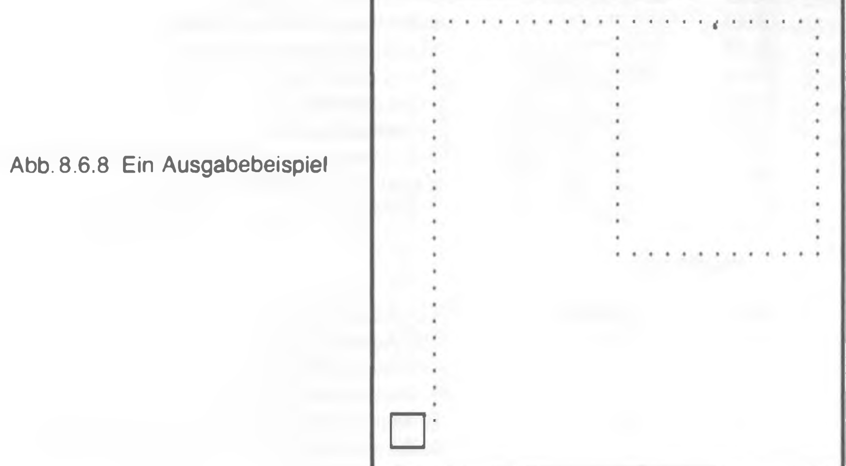


Abb. 8.6.8 Ein Ausgabebeispiel

zu Abb. 8.6.9

Flomon–Einsprünge

F000	JP START	Start des Flomon-Programms
F003	JP CI	Ein Zeichen von der Tastatur lesen. Das Zeichen kommt in Register A an.
F006	JP RI	Ein Zeichen von der CAS-Schnittstelle lesen. Das Zeichen kommt in A an.
F009	JP CO	Ein Zeichen auf dem Bildschirm ausgeben. Das Zeichen steht dazu im Register C.
F00C	JP PO	Ein Zeichen auf die CAS-Schnittstelle ausgeben Das Zeichen steht dazu im Register C.
F00F	JP LO	Ein Zeichen auf die Druckerschnittstelle ausgeben. Das Zeichen steht im Register C.
F012	JP CSTS	Consolstatus prüfen. Im Register A steht der Wert 0FFh, wenn ein Zeichen ansteht, sonst der Wert 0. Das Zeichen wird erst durch CI tatsächlich gelesen.
F015	JP IOBYTE	Nicht implementiert. Jedoch aus komp. Gründen belassen, liefert den Wert 0 in A,
F018	JP IOSET	Nicht implementiert.

```
F01B    JP MEMCHK       A=EFh und B=FF, letzte verfügbare RAM-Zelle.
F01E    JP START        Neustart des Flomon.
F021    JP FLOPPY       Universal Floppy-Einsprung Mini/Maxi.
F024    JP MAXI         mit MAXI-Voreinstellung, mc-kompatibel.
F027    JP MINI         mit MINI-Voreinstellung, mc-kompatibel.
F02A    JP WINCHESTER   für spätere Erweiterung mit Winchester Laufwerk.
;
        DEFW 0          0=keine Tabelle vorhanden.
        DEFW LASTMON    letzte Speicherzelle im Monitorprogramm
        DEFW FREEMEM    dort sicher Platz fuer eigene Buffer.
        DEFS 13         Reserve fuer Erweiterungen
; Grafik-Befehle für schnelle Grafiken
;
F040    JP CLRALL       alle Bildseiten löschen
F043    JP CLRINVIS     unsichtbare Bildseite löschen
F046    JP MOVETO       HL=X,DE=Y, Positionieren
F049    JP DRAWTO       HL=X,DE=Y, Linie zeichnen
F04C    JP WRTPAGE      C=Schreibseite 0,1,2,3
F04F    JP VIEWPAGE     C=Leseseite 0,1,2,3
F052    JP RMWPAGE      C=0, normal, =1 mit XOR-Mode, nur mit Zusatzhardw.
F055    JP WAIT         Wartet bis der GDP fertig ist
F058    JP CMD          C=Befehl, wird an den GDP ausgegeben
;
; Spezialbefehle
;
F05B    JP BANK         HL=Adresse in der Quellbank 0..EFFF
                        DE=Adresse in der Zielbank  0..EFFF
                        C=Nummer der Quellbank      0..F
                        B=Nummer der Zielbank       0..F
                        Es werden 128 Bytes transportiert.
                        Dabei muß von Adresse F000 bis FFFF
                        auf der jeweiligen Bank auf jeden Fall
                        ein RAM-Speicher sein. Sonst wird
                        ein Carry als Ergebnis geliefert.
                        Der Bereich F000 bis FFFF ist immer
                        für das Transportprogramm reserviert.
```

Abb. 8.6.9 Die Flomon-Einsprünge

Abb. 8.6.1 Hexdump des Flomon 4.0 zu Abb. 8.6.10

```
------------- Seite  1 ------- Datei Flomon 4.0 -----------
Rom                                                     Checksumme
0000  C3 BA 02 79 E6 0F D3 C8 3E 55 32 9E F5 3A 9E F5   += 08AD
0010  FE 55 37 20 65 3E AA 32 9E F5 3A 9E F5 FE AA 37   += 0868
0020  20 58 D9 21 7F 00 11 9E F5 01 1F 00 ED B0 D9 C3   += 06EE
0030  9E F5 D9 21 1E F6 11 44 68 01 80 00 ED B0 D9 78   += 07CD
0040  E6 0F D3 C8 3E 55 32 9E F5 3A 9E F5 FE 55 37 20   += 085F
0050  29 3E AA 32 9E F5 3A 9E F5 FE AA 37 20 1C D9 21   += 07B8
0060  44 68 11 1E F6 01 80 00 ED B0 21 9E 00 11 9E F5   += 0652
0070  01 1F 00 ED B0 D9 C3 9E F5 AF 3E 00 D3 C8 C9 79   += 08B6
```

```
0080  E6 0F F6 80 D3 C8 7C D9 67 D9 7D D9 6F 11 1E F6    += 0985
0090  01 80 00 ED B0 D9 79 E6 0F D3 C8 C3 32 00 78 E6    += 0853
00A0  0F F6 80 D3 C8 7A D9 57 D9 7B D9 5F 21 1E F6 01    += 088C
00B0  80 00 ED B0 D9 78 E6 0F D3 C8 C3 79 00 3A 33 F0    += 0897
00C0  CB 77 C2 F9 01 3A 81 60 B7 20 1C 3A 38 68 FE 64    += 0748
00D0  C2 E3 00 E5 D5 C5 CD 7A 01 C1 D1 E1 AF 32 38 68    += 0960
00E0  C3 E7 00 3C 32 38 68 CD 1C 01 DB 68 CB 7F C2 F7    += 07E8
00F0  00 3E FF B7 C3 F8 00 AF C9 3A 33 F0 CB 77 C2 08    += 0890
0100  02 E5 D5 C5 CD 7A 01 CD 1C 01 DB 68 CB 7F C2 07    += 0809
0110  01 F5 CD B2 01 DB 69 F1 C1 D1 E1 C9 3A 5C 60 B7    += 0994
0120  CA 4A 01 CD 6D 05 DA 4A 01 3A 5B 60 FE 01 C2 46    += 0675
0130  01 3A 5C 60 32 5B 60 3A 84 60 3C E6 03 32 84 60    += 053D
0140  CD 02 04 C3 4A 01 3D 32 5B 60 3A 5E 60 B7 CA 79    += 05FD
0150  01 CD 6D 05 DA 79 01 3A 5D 60 FE 01 C2 75 01 3A    += 05FC
0160  5E 60 32 5D 60 3A 84 60 EE 01 E6 03 32 84 60 CD    += 0686
0170  02 04 C3 79 01 3D 32 5D 60 C9 3A 0F 60 FE 01 C2    += 05A2
0180  AC 01 CD 89 03 DB 71 F5 3A 85 60 F5 CD D3 11 E5    += 08F1
0190  D5 CD 25 04 CD 50 05 CD 79 07 D1 E1 CD 4E 04 F1    += 07FC
01A0  32 85 60 CD 02 04 CD 89 03 F1 D3 71 3E 01 32 81    += 066A
01B0  60 C9 3A 0F 60 FE 01 C2 E1 01 CD 89 03 DB 71 F5    += 080F
01C0  3A 85 60 F5 CD D3 11 E5 D5 CD 53 05 CD 25 04 CD    += 0867
01D0  79 07 D1 E1 CD 4E 04 F1 32 85 60 CD 02 04 F1 D3    += 07F0
01E0  71 AF 32 81 60 C9 3A 33 F0 E6 0F C2 F0 01 3E 0E    += 074D
01F0  F6 10 D3 F3 3E 0B D3 F2 C9 DB F1 CB 5F C2 04 02    += 0961
0200  AF C3 07 02 3E FF B7 C9 DB F1 CB 5F 28 FA DB F0    += 0A1B
0210  C9 DB F1 CB 77 20 FA CB 67 28 F6 79 D3 F0 C9 3A    += 0A80
0220  33 F0 CB 67 C2 54 03 DB 49 0F 38 FB 79 D3 48 AF    += 0817
0230  D3 49 3E 01 D3 49 79 C9 DB 69 32 33 F0 AF 32 38    += 076B
0240  68 32 85 60 32 84 60 32 39 68 32 81 60 32 86 60    += 0593
0250  32 87 60 32 5F 60 32 59 60 D3 A0 32 45 60 32 60    += 05D1
0260  60 32 7E 60 32 6B 60 32 77 60 32 7B 60 32 7F 60    += 0594
0270  32 58 60 3E 01 32 80 60 32 7A 60 21 B9 6A 22 42    += 04EF
0280  68 21 FF 6F 22 3E 68 21 00 00 22 00 60 AF 32 7E    += 04C1
0290  60 32 7F 60 3E 00 32 5E 60 32 5D 60 3A 33 F0 E6    += 05D1
02A0  D0 FE C0 CA B3 02 3E 07 CD 90 03 3E 02 CD 90 03    += 0752
02B0  CD BD 03 CD E6 01 CD 32 02 C9 31 02 F7 21 71 18    += 06DF
02C0  11 00 F0 01 FF 0F ED B0 CD 38 02 31 02 F7 21 E7    += 06E6
02D0  02 E5 C5 D5 E5 D9 C5 D5 E5 DD E5 FD E5 ED 73 3A    += 0BFC
02E0  68 31 B8 6A C3 EA 02 C3 00 F0 3E 01 32 0F 60 E5    += 06E2
02F0  21 1A 03 E3 CD 1B 03 E6 7F FE 1B C2 04 03 CD 63    += 0683
0300  14 C3 17 03 FE 20 D2 0F 03 CD 21 17 C3 17 03 FE    += 05D3
0310  20 DA 17 03 CD 06 18 C3 F4 C2 C9 3A 59 60 FE 01    += 0673
0320  C2 27 03 CD F9 00 C9 C5 D5 E5 D9 C5 D5 E5 DD E5    += 0B14
0330  FD E5 ED 73 3C 68 ED 7B 3A 68 C3 77 03 FD E1 DD    += 09E8
0340  E1 E1 D1 C1 D9 E1 D1 C1 37 3F C9 00 3A 33 F0 CB    += 0A07
0350  7F C2 11 02 79 C5 D5 E5 D9 C5 D5 E5 DD E5 FD E5    += 0B48
0360  F5 3A 81 60 B7 CA 6B 03 CD B2 01 F1 ED 73 3A 68    += 0872
0370  ED 7B 3C 68 C3 3D 03 CD 1C 01 FD E1 DD E1 E1 D1    += 0947
0380  C1 D9 E1 D1 C1 37 3F 79 C9 DB 70 E6 04 28 FA C9    += 09E5
0390  F5 CD 89 03 F1 D3 70 C9 CD 89 03 18 0F F5 CD 89    += 0916
03A0  03 F1 E6 F0 47 DB 60 E6 0F B0 D3 60 3E 07 D3 71    += 08AD
03B0  3E 04 CD 90 03 CD 89 03 3E 03 D3 71 C9 3E 00 CD    += 0654
```

zu Abb. 8.6.10

```
03C0  9D 03 3E 50 CD 9D 03 3E A0 CD 9D 03 3E F0 CD 9D    += 077E
03D0  03 21 B8 60 11 B9 60 01 7F 07 36 20 ED B0 AF 32    += 05C1
03E0  82 60 32 83 60 21 A0 60 0E 00 06 18 71 0C 23 05    += 03E9
03F0  C2 EC 03 21 88 60 0E 00 06 18 36 00 0C 23 05 C2    += 0412
0400  FA 03 C5 3A 85 60 07 07 E6 0C 47 3A 84 60 E6 03    += 062F
0410  B0 07 07 07 07 47 3A 86 60 E6 0F B0 C1 F5 CD 89    += 06E4
0420  03 F1 D3 60 C9 E5 D5 3A 82 60 4F 69 06 00 60 29    += 070D
0430  09 29 E5 3A 83 60 4F 69 06 00 60 29 29 09 29 11    += 03E7
0440  F6 00 EB AF ED 52 EB E1 CD 4E 04 D1 E1 C9 CD 89    += 0A8B
0450  03 7C D3 78 7D D3 79 7A D3 7A 7B D3 7B C9 3A 60    += 0886
0460  60 FE 01 C2 94 04 E5 D5 CD 17 09 CD D3 11 22 67    += 079A
0470  60 ED 53 69 60 D1 E1 E5 D5 CD 98 04 CD 2C 09 2A    += 086A
0480  67 60 ED 5B 69 60 CD 4E 04 D1 E1 CD 98 04 CD 17    += 07F6
0490  09 C3 97 04 CD 98 04 C9 22 02 60 ED 53 04 60 CD    += 068E
04A0  89 03 DB 78 CB 5F 28 02 F6 F0 57 DB 79 5F ED 53    += 0863
04B0  06 60 2A 02 60 AF ED 52 22 0A 60 DB 7A CB 5F 28    += 0613
04C0  02 F6 F0 57 DB 7B 5F ED 53 08 60 2A 04 60 AF ED    += 07C6
04D0  52 22 0C 60 ED 5B 0A 60 CB 7C 28 03 CD DC 10 CB    += 0688
04E0  7A 28 05 EB CD DC 10 EB 7C B7 C2 F2 04 7A B7 CA    += 091C
04F0  21 05 2A 02 60 ED 5B 04 60 E5 D5 2A 0A 60 CB 2C    += 05A3
0500  CB 1D ED 5B 06 60 19 E5 2A 0C 60 CB 2C CB 1D ED    += 06F6
0510  5B 08 60 19 EB E1 CD 98 04 D1 E1 CD 98 04 C3 2B    += 081A
0520  05 2A 0A 60 ED 5B 0C 60 CD 2C 05 C9 06 11 CB 7C    += 0572
0530  28 05 CD DC 10 CB C8 CB 7A 28 07 EB CD DC 10 EB    += 087C
0540  CB D0 CD 89 03 7D D3 75 7B D3 77 78 CD 90 03 C9    += 091F
0550  AF 18 02 3E 01 CD 90 03 3E 02 C3 90 03 F5 CD 89    += 0649
0560  03 F1 D3 72 C9 F5 CD 89 03 F1 D3 73 C9 DB 70 E6    += 0A81
0570  02 CA 8A 05 3A 7E 60 B7 C2 85 05 3C 32 7E 60 37    += 05F9
0580  3F C9 C3 87 05 37 C9 C3 9C 05 3A 7E 60 B7 C2 96    += 07E2
0590  05 37 C9 C3 9C 05 AF 32 7E 60 37 C9 C9 3A 83 60    += 070E
05A0  47 CD 46 06 3A 82 60 5F 16 00 19 C9 B7 F2 B6 05    += 0637
05B0  CD BE 07 C3 B9 05 CD 90 03 C9 CD 53 05 AF CD 65    += 0842
05C0  05 AF CD 5D 05 11 00 00 06 04 21 00 00 CD 4E 04    += 033E
05D0  0E 08 3E 0B CD 90 03 0D C2 D2 05 21 40 00 19 EB    += 04CA
05E0  05 C2 CA 05 3E 11 C3 65 05 06 08 C5 3A A0 60 F5    += 0614
05F0  DD 21 A0 60 06 17 DD 7E 01 DD 77 00 DD 23 05 C2    += 0692
0600  F6 05 F1 DD 77 00 C1 05 C2 EB 05 C9 3A 85 60 32    += 07D2
0610  84 60 CD 02 04 CD 98 03 CD 50 05 21 00 00 11 F6    += 0569
0620  00 CD 4E 04 CD BA 06 CD E9 05 0E 10 C5 41 CD 46    += 069E
0630  06 AF 02 54 5D 13 01 4F 00 36 20 ED B0 C1 0C 79    += 0504
0640  FE 18 C2 2C 06 C9 D5 58 16 00 21 A0 60 19 5E 21    += 05CF
0650  88 60 19 44 4D 62 6B 29 29 29 29 54 5D 29 29 19    += 041F
0660  11 B8 60 19 D1 C9 3E 4F 32 7C 60 18 05 3E 50 32    += 0554
0670  7C 60 CD 9D 05 3A 82 60 4F DB 70 E6 04 CA 79 06    += 0734
0680  7E B7 F2 8B 06 CD BE 07 C3 8D 06 D3 70 23 0C 3A    += 074C
0690  7C 60 B9 C2 79 06 C9 3E 17 32 7D 60 18 05 3E 18    += 0576
06A0  32 7D 60 CD 9D 05 0A 32 54 60 3A 83 60 47 C5 3A    += 05D1
06B0  54 60 47 3A 82 60 4F C3 DA 06 3E 18 32 7D 60 06    += 0574
06C0  08 C5 CD 46 06 0A 47 0E 00 18 0F 3E 18 32 7D 60    += 03D1
06D0  06 00 C5 CD 46 06 0A 47 0E 00 78 B7 CA FA 06 DB    += 0617
06E0  70 E6 04 CA DF 06 7E B7 F2 F1 06 CD BE 07 C3 F3    += 096F
06F0  06 D3 70 23 0C 79 B8 C2 DF 06 CD 89 03 AF D3 78    += 07A3
```

zu Abb. 8.6.10

```
0700  D3 79 DB 7A 67 DB 7B 6F 11 0A 00 AF ED 52 7C D3    += 0825
0710  7A 7D D3 7B C1 04 3A 7D 60 B8 C2 D2 06 C9 5B 5C    += 07F3
0720  5D 7B 7C 7D 7E 40 DB DC DD FB FC FD FE C0 7D 0A    += 0A5C
0730  09 0A 7D 3D 42 42 42 3D 7D 40 40 40 7D 71 54 54    += 04A3
0740  78 41 00 39 44 44 39 3D 40 40 7D 40 00 7F 01 4D    += 03FA
0750  32 00 4A 55 29 00 4F 3A 39 68 B7 28 04 79 F6 80    += 04F6
0760  4F 3A 58 60 B7 20 02 79 C9 79 21 1E 07 01 10 00    += 042C
0770  ED B1 C0 3E 0F 91 F6 80 C9 CD 89 03 DB 71 F5 3E    += 0953
0780  03 CD 90 03 3E AC CD 90 03 CD 89 03 F1 D3 71 4F    += 078A
0790  3E F8 CD 90 03 3E A8 CD 90 03 DB 71 F5 3E 03 CD    += 082B
07A0  90 03 3E FE CD 90 03 3E AE CD 90 03 CD 89 03 F1    += 07C5
07B0  D3 71 79 C9 CD 8F 07 CD 90 03 E1 D1 C1 C9 C5 D5    += 0A1F
07C0  E5 E6 7F FE 20 28 F0 FE 21 30 E9 FE 08 38 05 CD    += 08C8
07D0  8F 07 D6 08 5F 16 00 62 6B 29 29 19 11 2E 07 19    += 0380
07E0  CD AF 11 D9 D5 E5 2A 25 60 ED 5B 27 60 D9 06 05    += 0782
07F0  7E D9 D5 47 0E 08 CB 00 D2 00 08 3E 80 CD 90 03    += 064C
0800  13 CD 4E 04 0D C2 F6 07 D1 23 CD 4E 04 D9 23 05    += 0612
0810  C2 F0 07 D9 23 CD 4E 04 E1 D1 D9 E1 D1 C1 C9 CD    += 0A68
0820  2E 08 D2 2B 08 E6 7F 37 C3 2D 08 E6 7F C9 CD 1B    += 06E5
0830  03 C9 AF 32 5A 60 CD 1F 08 FE 20 CA 36 08 FE 25    += 06A4
0840  C2 51 08 CD 1F 08 E5 CD 55 08 D1 F5 19 F1 C3 54    += 0805
0850  08 CD 55 08 C9 21 00 00 FE 2D C2 65 08 3E 01 32    += 04E7
0860  5A 60 CD 1F 08 FE 24 C2 DD 08 CD 1F 08 FE 3A 30    += 06D3
0870  05 FE 30 D2 8A 08 FE 47 30 05 FE 41 D2 8A 08 FE    += 07B2
0880  67 30 05 FE 61 D2 8A 08 37 C9 FE 3A 30 05 FE 30    += 06FA
0890  D2 A6 08 FE 47 30 05 FE 41 D2 A6 08 FE 67 D2 DA    += 08CA
08A0  08 FE 61 DA DA 08 29 29 29 29 FE 47 D2 BB 08 FE    += 079F
08B0  41 DA BB 08 D6 41 C6 0A C3 CE 08 FE 67 D2 CC 08    += 0869
08C0  FE 61 DA CC 08 D6 61 C6 0A C3 CE 08 D6 30 E6 0F    += 08A8
08D0  5F 16 00 19 CD 1F 08 C3 8A 08 C3 04 09 FE 3A 30    += 050F
08E0  05 FE 30 D2 E8 08 37 C9 FE 3A D2 04 09 FE 30 DA    += 0814
08F0  04 09 54 5D 29 29 19 29 E6 0F 5F 16 00 19 CD 1F    += 03C1
0900  08 C3 E8 08 F5 3A 5A 60 FE 00 CA 13 09 DA 13 09    += 067E
0910  CD DC 10 F1 37 3F C9 3A 61 60 32 85 60 CD 02 04    += 06CE
0920  3A 62 60 D3 A0 3A 63 60 CD 90 03 C9 3A 64 60 32    += 06C5
0930  85 60 CD 02 04 3A 65 60 D3 A0 3A 66 60 CD 90 03    += 068A
0940  C9 3A 5F 60 B7 C2 4E 09 32 5E 60 32 5C 60 CD 50    += 068D
0950  05 CD 1F 08 FE 41 CA 92 0E FE 4D C2 7C 09 CD AF    += 07B0
0960  11 2A 25 60 CD 32 08 DA 79 09 E5 2A 27 60 CD 32    += 05B8
0970  08 D1 DA 79 09 EB CD 4E 04 C3 8C 0E FE 44 C2 9F    += 083F
0980  09 CD AF 11 2A 25 60 CD 32 08 DA 9C 09 E5 2A 27    += 0601
0990  60 CD 32 08 D1 DA 9C 09 EB CD 0B 13 C3 8C 0E FE    += 07E8
09A0  50 C2 BE 09 CD 32 08 DA BB 09 7D E6 03 32 84 60    += 06FA
09B0  7D 0F 0F E6 03 32 85 60 CD 02 04 C3 8C 0E FE 53    += 061C
09C0  C2 E3 09 CD 32 08 DA E0 09 7D E6 03 32 84 60 7D    += 0771
09D0  0F 0F E6 03 32 85 60 CD 6D 05 DA D7 09 CD 02 04    += 05EA
09E0  C3 8C 0E FE 72 C2 FE 09 CD 32 08 DA F4 09 7D E6    += 08D7
09F0  0F C3 F5 09 AF 32 86 60 CD 02 04 C3 8C 0E FE 59    += 071E
0A00  C2 1F 0A CD 32 08 DA 18 0A 7D 32 5E 60 32 5D 60    += 054A
0A10  3E 01 32 5F 60 C3 1C 0A AF 32 5F 60 C3 8C 0E FE    += 0614
0A20  58 C2 40 0A CD 32 08 DA 39 0A 7D 32 5C 60 32 5B    += 0580
0A30  60 3E 01 32 5F 60 C3 3D 0A AF 32 5F 60 C3 8C 0E    += 0597
```

zu Abb. 8.6.10

```
0A40  FE 43 C2 4E 0A CD BA 05 CD 50 05 C3 8C 0E FE 5A    += 07BE
0A50  C2 5C 0A CD BD 03 CD 50 05 C3 8C 0E FE 47 C2 8D    += 07C8
0A60  0A CD 32 08 DA 8A 0A E5 CD 32 08 D1 DA 8A 0A 7B    += 0725
0A70  FE 10 D2 8A 0A FE 00 DA 8A 0A 7C B7 C2 8A 0A CD    += 0836
0A80  89 03 7B E6 0F F6 70 4F ED 69 C3 8C 0E FE 42 C2    += 0866
0A90  E4 0A CD 1F 08 FE 0D CA E1 0A F5 3A 60 60 FE 01    += 0790
0AA0  C2 CD 0A F1 FE 80 D2 CA 0A FE 20 DA CA 0A 4F CD    += 0996
0AB0  17 09 CD D3 11 E5 D5 79 CD 90 03 D1 E1 CD 2C 09    += 0818
0AC0  CD 4E 04 79 CD 90 03 CD 17 09 C3 DB 0A F1 FE 80    += 07FC
0AD0  D2 DB 0A FE 20 DA DB 0A CD 90 03 CD 1F 08 C3 95    += 0840
0AE0  0A C3 8C 0E FE 56 C2 FC 0A CD 2E 08 B7 CA F9 0A    += 080A
0AF0  CD 90 03 CD 2E 08 C3 EC 0A C3 8C 0E FE 6D C2 1D    += 07C3
0B00  0B CD 2E 08 67 E5 CD 2E 08 E1 6F E5 CD 2E 08 57    += 06EC
0B10  D5 CD 2E 08 D1 5F E1 CD 4E 04 C3 8C 0E FE 64 C2    += 0889
0B20  3E 0B CD 2E 08 67 E5 CD 2E 08 E1 6F E5 CD 2E 08    += 06D3
0B30  57 D5 CD 2E 08 D1 5F E1 CD 5E 04 C3 8C 0E FE 4F    += 0819
0B40  C2 87 0B 21 00 00 CD 32 08 22 15 60 21 00 00 CD    += 0401
0B50  32 08 22 17 60 21 00 00 CD 32 08 22 11 60 21 00    += 02AF
0B60  00 CD 32 08 22 13 60 FE 20 C2 75 0B 21 00 00 CD    += 04EA
0B70  32 08 C3 76 0B 37 D2 7D 0B AF C3 7E 0B 7D 32 43    += 05FC
0B80  60 CD EF 11 C3 8C 0E FE 52 C2 3F 0C 21 00 00 CD    += 06D5
0B90  32 08 22 31 60 21 00 00 CD 32 08 22 33 60 FE 20    += 03E8
0BA0  C2 AC 0B 21 00 00 CD 32 08 C3 AD 0B 37 DA 0D 0C    += 0546
0BB0  7D FE 01 C2 0A 0C CD AF 11 2A 31 60 7C B5 CA CF    += 0766
0BC0  0B CB 7C CA CC 0B 21 FF FF C3 CF 0B 21 01 00 22    += 06F3
0BD0  35 60 2A 25 60 ED 5B 27 60 CD 4E 04 EB ED 4B 33    += 0688
0BE0  60 09 EB E5 CD 5E 04 E1 ED 5B 35 60 19 E5 EB 2A    += 0839
0BF0  25 60 ED 4B 31 60 09 AF ED 52 7C B5 E1 C2 D5 0B    += 07F9
0C00  2A 25 60 ED 5B 27 60 CD 4E 04 C3 3C 0C CD AF 11    += 0635
0C10  2A 27 60 ED 4B 33 60 09 EB 2A 25 60 E5 D5 CD 5E    += 0704
0C20  04 D1 E1 ED 4B 31 60 09 E5 CD 5E 04 E1 ED 5B 27    += 07EC
0C30  60 D5 CD 5E 04 D1 2A 25 60 CD 5E 04 C3 8C 0E FE    += 076E
0C40  4C C2 8B 0C CD AF 11 2A 25 60 CD 32 08 E5 2A 27    += 061E
0C50  60 CD 32 08 EB E1 CD 4E 04 CD AF 11 E5 21 7E 0C    += 076F
0C60  E3 2A 25 60 CD 32 08 FE 20 C0 E5 2A 27 60 CD 32    += 070C
0C70  08 D1 EB F5 CD 5E 04 F1 FE 20 C0 C3 61 0C 2A 25    += 0836
0C80  60 ED 5B 27 60 CD 5E 04 C3 8C 0E FE 6C C2 96 0C    += 0789
0C90  CD 46 13 C3 8C 0E FE 4A C2 C7 0C 21 00 00 CD 32    += 0680
0CA0  08 22 31 60 21 00 00 CD 32 08 22 33 60 CD AF 11    += 0425
0CB0  2A 27 60 ED 4B 33 60 09 EB 2A 25 60 ED 4B 31 60    += 05E8
0CC0  09 CD 0B 13 C3 8C 0E FE 57 C2 D2 0C CD 9D 0E C3    += 0781
0CD0  8C 0E FE 46 C2 7B 0D 3A 86 60 F5 CD AF 11 2A 25    += 0719
0CE0  60 CD 32 08 DA 5E 0D E5 2A 27 60 CD 32 08 D1 DA    += 06F4
0CF0  5B 0D ED 53 71 60 22 73 60 21 00 00 CD 32 08 DA    += 0570
0D00  5B 0D E5 FE 20 C2 12 0D CD 32 08 DA 12 0D 7D 32    += 05FB
0D10  86 60 E1 3A 86 60 32 87 60 3A 85 60 F5 7D E6 03    += 077A
0D20  32 75 60 CD 94 0F DA 54 0D CD CE 0F ED 5B 73 60    += 0777
0D30  ED 53 6E 60 2A 71 60 22 6C 60 CD 4E 04 3A 75 60    += 0625
0D40  32 70 60 32 85 60 CD 02 04 CD 50 05 CD 08 10 3E    += 0531
0D50  01 32 6B 60 F1 32 85 60 CD 02 04 C3 71 0D 3A 87    += 05DB
0D60  60 32 86 60 CD 02 04 CD CE 0F AF 32 6B 60 CD 50    += 06BE
0D70  05 F1 32 86 60 CD 02 04 C3 8C 0E FE 45 C2 83 0D    += 06D3
```

zu Abb. 8.6.10

```
0D80  C3 8C 0E FE 48 C2 A3 0D 21 00 00 CD 32 08 DA 9C    += 06B3
0D90  0D 22 39 60 3E 01 32 45 60 C3 A0 0D AF 32 45 60    += 04D4
0DA0  C3 8C 0E FE 49 C2 C3 0D 21 00 00 CD 32 08 DA BC    += 06F4
0DB0  0D 22 37 60 3E 02 32 45 60 C3 C0 0D AF 32 45 60    += 04F3
0DC0  C3 8C 0E FE 4B C2 1E 0E 21 00 00 CD 32 08 DA 17    += 05AD
0DD0  0E 7D E6 03 32 61 60 21 00 00 CD 32 08 7D 32 62    += 04A0
0DE0  60 21 00 00 CD 32 08 7D E6 01 32 63 60 21 00 00    += 0402
0DF0  CD 32 08 7D E6 03 32 64 60 21 00 00 CD 32 08 7D    += 0508
0E00  32 65 60 21 00 00 CD 32 08 7D E6 01 32 66 60 3E    += 04B9
0E10  01 32 60 60 C3 1B 0E AF 32 60 60 C3 8C 0E FE 51    += 062C
0E20  C2 2C 0E 21 FF 6F 22 3E 68 C3 8C 0E FE 54 C2 49    += 070D
0E30  0E CD D3 11 ED 73 40 68 ED 7B 3E 68 E5 D5 ED 73    += 08EF
0E40  3E 68 ED 7B 40 68 C3 8C 0E FE 55 C2 74 0E 2A 3E    += 0712
0E50  68 11 FF 6F AF ED 52 7C B5 CA 71 0E ED 73 40 68    += 0857
0E60  ED 7B 3E 68 D1 E1 ED 73 3E 68 ED 7B 40 68 CD 4E    += 08F1
0E70  04 C3 8C 0E FE 1B C2 8C 0E CD 1F 08 FE 1B C2 8C    += 0731
0E80  0E CD 1F 08 FE 41 C2 8C 0E C3 92 0E CD 1F 08 C3    += 06B7
0E90  54 09 CD 89 03 3E 11 D3 73 AF D3 72 C9 CD 1F 08    += 06FC
0EA0  FE 41 C2 37 0F CD CE 0F AF 32 6B 60 CD 1F 08 FE    += 078F
0EB0  60 30 05 FE 40 D2 C2 0E FE 30 CA C2 0E FE 31 C2    += 082E
0EC0  22 0F 21 B9 6A 22 78 60 FE 30 C2 D3 0E 3E 03 77    += 05F8
0ED0  C3 FB 0E FE 31 C2 DE 0E 3E 02 77 C3 FB 0E E6 1F    += 0831
0EE0  47 DD 21 84 0F E6 07 5F 16 00 DD 19 78 E6 18 DD    += 0683
0EF0  B6 00 47 07 07 E6 60 F6 80 B0 77 23 E5 CD 1F 08    += 06EA
0F00  E1 FE 60 30 05 FE 40 D2 C8 0E FE 30 CA C8 0E FE    += 0926
0F10  31 CA C8 0E 36 00 23 3E 01 32 77 60 22 42 68 C3    += 0501
0F20  34 0F 3A 77 60 FE 01 C2 34 0F AF 32 77 60 2A 78    += 05B2
0F30  60 22 42 68 C3 83 0F FE 42 C2 42 0F CD 08 10 C3    += 067C
0F40  83 0F FE 43 C2 75 0F CD CE 0F AF 32 6B 60 CD 32    += 076E
0F50  08 DA 69 0F E5 CD 32 08 D1 DA 66 0F 7B 32 7A 60    += 06ED
0F60  7D E6 07 32 7B 60 C3 72 0F 3E 01 32 7A 60 AF 32    += 05E7
0F70  7B 60 C3 83 0F FE 47 C2 83 0F 21 00 00 CD 32 08    += 05F1
0F80  7D D3 A0 C9 00 01 02 03 06 07 04 05 01 02 03 06    += 02E1
0F90  05 00 07 04 3A 6B 60 B7 C2 9D 0F AF C9 2A 6C 60    += 05A8
0FA0  ED 5B 71 60 AF ED 52 7C B5 CA AE 0F AF C9 2A 6E    += 08CF
0FB0  60 ED 5B 73 60 AF ED 52 7C B5 CA BF 0F AF C9 3A    += 08E4
0FC0  70 60 47 3A 75 60 B8 CA CC 0F AF C9 37 C9 3A 6B    += 07A0
0FD0  60 FE 01 C2 07 10 ED 5B 6E 60 2A 6C 60 CD 4E 04    += 0663
0FE0  3A 85 60 F5 3A 70 60 32 85 60 CD 02 04 3A 86 60    += 0628
0FF0  B7 C2 FA 0F CD 53 05 C3 FD 0F CD 50 05 CD 08 10    += 077D
1000  F1 32 85 60 CD 02 04 C9 3A 77 60 FE 01 C2 5D 10    += 06E3
1010  3A 7A 60 47 3A 7B 60 4F 2A 78 60 7E B7 CA 5A 10    += 062A
1020  C5 7E FE 80 DA 4B 10 79 B7 CA 47 10 7E E6 07 DD    += 088F
1030  21 8C 0F 5F 16 00 DD 19 DD 7E 00 0D C2 2F 10 4F    += 04DF
1040  7E E6 F8 B1 C3 48 10 7E C3 4C 10 7E 23 F5 CD 90    += 08B8
1050  03 F1 05 C2 4D 10 C1 C3 1B 10 C3 60 10 CD 6F 10    += 0646
1060  C9 03 18 02 1E 1E 03 18 1A 02 1C 1C 03 1A 02 CD    += 027D
1070  89 03 3E 14 D3 75 3E 14 D3 77 06 0E 21 61 10 7E    += 04E6
1080  CD 90 03 23 10 F9 C9 11 68 01 7C B7 FA A3 10 E5    += 0794
1090  21 A0 10 E3 E5 AF ED 52 E1 F8 AF ED 52 C3 94 10    += 09B5
10A0  C3 A9 10 19 7C B7 FA A3 10 C9 0E 00 7A B7 F2 B9    += 0828
10B0  10 0E 01 2F 57 7B 2F 5F 13 78 A9 32 10 60 CB 44    += 0493
```

zu Abb. 8.6.10

```
10C0  C2 E4 10 7D 0E 08 21 00 00 29 17 D2 D1 10 19 CE    += 0544
10D0  00 0D C2 C9 10 6C 67 3A 10 60 B7 C8 7C 2F 67 7D    += 0633
10E0  2F 6F 23 C9 EB C3 D7 10 00 04 09 0D 12 16 1B 1F    += 049B
10F0  24 28 2C 31 35 3A 3E 42 47 4B 4F 53 58 5C 60 64    += 0444
1100  68 6C 70 74 78 7C 80 84 88 8B 8F 93 96 9A 9E A1    += 0854
1110  A5 A8 AB AF B2 B5 B8 BB BE C1 C4 C7 CA CC CF D2    += 0BC2
1120  D4 D7 D9 DB DE E0 E2 E4 E6 E8 EA EC ED EF F1 F2    += 0E46
1130  F3 F5 F6 F7 F8 F9 FA FB FC FD FE FE FF FF FF 00    += 0EAD
1140  00 00 00 EB 21 5A 00 AF ED 52 CB 7C CA 53 11 11    += 05DA
1150  68 01 19 22 2F 60 06 00 11 5A 00 AF ED 52 FA 9B    += 0527
1160  11 11 5A 00 AF ED 52 FA 8E 11 06 01 11 5A 00 AF    += 0524
1170  ED 52 FA 82 11 21 68 01 ED 5B 2F 60 AF ED 52 C3    += 07DE
1180  8B 11 2A 2F 60 11 B4 00 AF ED 52 C3 98 11 ED 5B    += 06BC
1190  2F 60 21 B4 00 AF ED 52 C3 9E 11 2A 2F 60 7C B5    += 06AE
11A0  C8 EB 21 E8 10 19 6E 26 00 7D B7 C0 26 01 C9 CD    += 072A
11B0  89 03 DB 78 CB 5F 28 02 F6 F0 32 26 60 DB 79 32    += 0757
11C0  25 60 DB 7A CB 5F 28 02 F6 F0 32 28 60 DB 7B 32    += 0756
11D0  27 60 C9 CD 89 03 DB 78 CB 5F 28 02 F6 F0 67 DB    += 0878
11E0  79 6F DB 7A CB 5F 28 02 F6 F0 57 DB 7B 5F C9 CD    += 0919
11F0  AF 11 2A 13 60 CD 87 10 22 13 60 2A 11 60 CD 87    += 0545
1200  10 22 11 60 CD 43 11 ED 5B 15 60 CD AA 10 EB 2A    += 061D
1210  25 60 AF ED 52 22 19 60 22 29 60 2A 11 60 CD 53    += 0574
1220  11 ED 5B 17 60 CD AA 10 EB 2A 27 60 AF ED 52 22    += 0703
1230  1B 60 22 2B 60 2A 11 60 22 2D 60 2A 2B 60 22 1F    += 0368
1240  60 2A 29 60 22 1D 60 2A 2D 60 CD 43 11 ED 5B 15    += 04E7
1250  60 CD AA 10 ED 5B 19 60 19 22 29 60 2A 2D 60 CD    += 05F0
1260  53 11 ED 5B 17 60 CD AA 10 ED 5B 1B 60 19 22 2B    += 05D3
1270  60 3A 43 60 FE 01 C2 8B 12 2A 2B 60 22 23 60 2A    += 051F
1280  29 60 22 21 60 CD 81 13 C3 A5 12 2A 19 60 E5 2A    += 05B9
1290  1B 60 E5 2A 29 60 ED 5B 2B 60 CD 0B 13 E1 22 1B    += 05EF
12A0  60 E1 22 19 60 2A 2D 60 23 CD 87 10 22 2D 60 E5    += 05AE
12B0  2A 13 60 D1 AF ED 52 7C B5 C2 3B 12 2A 2D 60 CD    += 0720
12C0  43 11 ED 5B 15 60 CD AA 10 ED 5B 19 60 19 22 29    += 05BD
12D0  60 2A 2D 60 CD 53 11 ED 5B 17 60 CD AA 10 ED 5B    += 06D6
12E0  1B 60 19 22 2B 60 3A 43 60 FE 01 C2 00 13 2A 2B    += 0447
12F0  60 22 23 60 2A 29 60 22 21 60 CD 81 13 C3 0A 13    += 049C
1300  2A 29 60 ED 5B 2B 60 CD 0B 13 C9 3A 45 60 B7 C2    += 0692
1310  18 13 CD 5E 04 C3 45 13 22 21 60 ED 53 23 60 E5    += 05C0
1320  D5 3A 45 60 32 44 60 2A 37 60 22 19 60 2A 39 60    += 04A9
1330  22 1B 60 CD D3 11 22 1D 60 ED 53 1F 60 CD 85 13    += 0611
1340  D1 E1 CD 4E 04 C9 CD AF 11 DD 21 19 60 06 03 DD    += 0784
1350  E5 C5 2A 25 60 CD 32 08 C1 DD E1 D8 DD 75 00 DD    += 08E6
1360  74 01 DD 23 DD 23 DD E5 C5 2A 27 60 CD 32 08 C1    += 0775
1370  DD E1 D8 DD 75 00 DD 74 01 DD 23 DD 23 05 C2 4F    += 0850
1380  13 AF 32 44 60 2A 21 60 ED 5B 1D 60 CD 4D 14 DD    += 0613
1390  2A 1D 60 D9 2A 23 60 ED 5B 1F 60 CD 4D 14 D9 FD    += 06F8
13A0  2A 1F 60 D5 E1 D5 D9 E1 AF ED 52 7C D9 B7 F2 BE    += 0A98
13B0  13 D9 D5 D9 E1 D5 EB 21 00 00 AF ED 52 D1 DD 22    += 091A
13C0  1D 60 FD 22 1F 60 E5 D5 C5 3A 44 60 B7 C2 DD 13    += 07E1
13D0  2A 19 60 ED 5B 1B 60 CD 4E 04 C3 FE 13 FE 01 C2    += 071A
13E0  EF 13 2A 1D 60 ED 5B 1B 60 CD 4E 04 C3 FE 13 FE    += 075D
13F0  02 C2 FE 13 2A 19 60 ED 5B 1F 60 CD 4E 04 2A 1D    += 05A5
```


zu Abb. 8.6.10

```
1400  60 ED 5B 1F 60 CD 5E 04 C1 D1 E1 C5 E5 2A 1D 60    += 081A
1410  ED 4B 21 60 AF ED 42 7C B5 E1 C1 28 1D 7C B7 FA    += 08DC
1420  32 14 DD 09 E5 D9 E1 AF ED 52 AF ED 52 E5 D9 E1    += 0A46
1430  18 8C D9 FD 09 D9 19 19 18 84 C5 E5 2A 1F 60 ED    += 076A
1440  4B 23 60 AF ED 42 7C B5 E1 C1 20 D1 C9 01 01 00    += 073B
1450  AF ED 52 F2 5C 14 CD DC 10 01 FF FF EB C9 C9 CD    += 0A52
1460  1B 03 C9 CD 5F 14 FE 2E C2 80 14 CD 5F 14 FE 35    += 071C
1470  D2 7D 14 FE 30 DA 7D 14 E6 0F 32 80 60 C3 91 15    += 076C
1480  FE 29 C2 8D 14 3E 01 32 39 68 C3 91 15 FE 28 C2    += 06ED
1490  99 14 AF 32 39 68 C3 91 15 FE 47 C2 A9 14 CD 5F    += 0788
14A0  14 3E 20 CD 06 18 C3 91 15 FE 3D C2 D5 14 CD 5F    += 06D8
14B0  14 D6 20 FE 18 D2 C0 14 FE 00 DA C0 14 32 83 60    += 0787
14C0  CD 5F 14 D6 20 FE 50 D2 D2 14 FE 00 DA D2 14 32    += 082C
14D0  82 60 C3 91 15 FE 3F C2 EB 14 3A 83 60 C6 20 4F    += 079B
14E0  3A 82 60 C6 20 4F 0E 0D C3 91 15 FE 44 C2 04 15    += 05F2
14F0  CD 5F 14 FE 4C C2 FD 14 3E 01 C3 FE 14 AF 32 59    += 07AB
1500  60 C3 91 15 FE 51 C2 0F 15 CD E4 15 C3 91 15 FE    += 082B
1510  57 C2 1A 15 CD 20 16 C3 91 15 FE 45 C2 25 15 CD    += 06C0
1520  64 16 C3 91 15 FE 52 C2 30 15 CD BF 16 C3 91 15    += 0745
1530  FE 54 CA 3A 15 FE 74 C2 40 15 CD 92 15 C3 91 15    += 07D1
1540  FE 59 CA 4A 15 FE 79 C2 50 15 CD B3 15 C3 91 15    += 081C
1550  FE 7A C2 69 15 CD 5F 14 FE 30 C2 61 15 AF C3 63    += 0833
1560  15 3E 01 32 58 60 C3 91 15 FE 1B C2 91 15 CD 5F    += 0654
1570  14 FE 47 C2 91 15 AF 32 0F 60 CD 41 09 3A 5F 60    += 0621
1580  B7 C2 8C 15 3E 00 32 5E 60 32 5D 60 3E 01 32 0F    += 04B7
1590  60 C9 CD 25 04 CD 53 05 CD 6D 06 CD 9D 05 3A 82    += 06AF
15A0  60 3C 02 3A 82 60 47 3E 50 90 47 36 20 23 05 C2    += 04A6
15B0  AB 15 C9 CD 25 04 CD 53 05 CD 9E 06 CD 9B 15 3A    += 06CC
15C0  83 60 C6 01 47 78 FE 18 D2 E3 15 C5 CD 46 06 AF    += 07D6
15D0  02 01 4F 00 E5 D1 13 36 20 ED B0 C1 78 C6 01 47    += 0655
15E0  C3 C5 15 C9 CD 25 04 CD 53 05 CD 6D 06 CD 25 04    += 06B7
15F0  CD 50 05 3E 20 CD 90 03 3A 82 60 FE 4F D2 03 16    += 0634
1600  CD 66 06 CD 9D 05 0A FE 50 D2 0E 16 3C 02 3A 82    += 05F0
1610  60 47 0E 20 7E 71 4F 23 04 78 FE 50 C2 14 16 C9    += 05B5
1620  CD 25 04 CD 53 05 CD 6D 06 CD 25 04 CD 50 05 3A    += 05AD
1630  82 60 F5 3C FE 50 D2 3F 16 32 82 60 CD 6D 06 F1    += 07CD
1640  32 82 60 CD 9D 05 3A 82 60 47 E5 DD E1 78 FE 4F    += 084E
1650  D2 5F 16 DD 7E 01 DD 77 00 DD 23 04 C3 4D 16 DD    += 06FE
1660  36 00 20 C9 AF 32 82 60 CD 53 05 CD 25 04 CD 9E    += 0668
1670  06 CD 50 05 3A 83 60 FE 17 D2 8B 16 F5 3C 32 83    += 06B3
1680  60 CD 25 04 F1 32 83 60 CD 97 06 21 A0 60 3A 83    += 06A4
1690  60 4F 06 00 09 3A 83 60 47 3A B7 60 4F 78 FE 18    += 0550
16A0  D2 AB 16 7E 71 4F 23 04 C3 9D 16 3A 83 60 47 CD    += 069F
16B0  46 06 AF 02 54 5D 13 01 4F 00 36 20 ED B0 C9 AF    += 057C
16C0  32 82 60 CD 53 05 CD 25 04 CD 9E 06 CD 50 05 CD    += 068F
16D0  25 04 3A 83 60 FE 17 D2 E6 16 F5 3C 32 83 60 CD    += 073C
16E0  9E 06 F1 32 83 60 21 A0 60 3A 83 60 4F 06 00 09    += 0546
16F0  7E F5 E5 DD E1 3A 83 60 47 78 FE 17 D2 0B 17 DD    += 08D8
1700  7E 01 DD 77 00 DD 23 04 C3 F9 16 F1 DD 77 00 06    += 06F4
1710  17 CD 46 06 AF 02 54 5D 13 01 4F 00 36 20 ED B0    += 04E8
1720  C9 FE 08 C2 4E 17 3A 82 60 B7 C2 47 17 3A 83 60    += 0706
1730  B7 CA 44 17 3E 4F 32 82 60 3A 83 60 B7 CA 44 17    += 0676
```

zu Abb. 8.6.10

```
1740  3D 32 83 60 C3 4B 17 3D 32 82 60 C3 05 18 FE 1A   += 05C0
1750  C2 65 17 CD BD 03 3E 00 32 5E 60 32 5D 60 AF 32   += 05C9
1760  5C 60 C3 05 18 FE 1E C2 74 17 AF 32 82 60 32 83   += 067D
1770  60 C3 05 18 FE 07 C2 7F 17 00 00 00 C3 05 18 FE   += 057B
1780  0B C2 92 17 3A 83 60 B7 CA 8F 17 3D 32 83 60 C3   += 06CF
1790  05 18 FE 0A C2 B1 17 3A 83 60 FE 17 D2 A6 17 3C   += 06AC
17A0  32 83 60 C3 AE 17 3E 10 32 83 60 CD 0C 06 C3 05   += 05A7
17B0  18 FE 16 C2 C5 17 3A 83 60 FE 17 D2 C2 17 3C 32   += 0715
17C0  83 60 C3 05 18 FE 0C CA CF 17 FE 09 C2 FC 17 3A   += 0793
17D0  82 60 FE 4F C2 F5 17 AF 32 82 60 3A 83 60 FE 17   += 07F2
17E0  C2 EE 17 3E 10 32 83 60 CD 0C 06 C3 F2 17 3C 32   += 0643
17F0  83 60 C3 F9 17 3C 32 82 60 C3 05 18 FE 0D C2 05   += 06B8
1800  18 AF 32 82 60 C9 E6 7F CD 56 07 F5 CD 25 04 CD   += 07EB
1810  9D 05 0A 5F 3A 82 60 3C BB CA 20 18 DA 20 18 02   += 0534
1820  7E 32 57 60 FE 20 CA 34 18 CD 53 05 3E 0A CD 90   += 0665
1830  03 CD 25 04 CD 50 05 F1 77 B7 F2 43 18 CD BE 07   += 0719
1840  C3 46 18 CD 90 03 3A 82 60 3C 32 82 60 FE 4F CA   += 0704
1850  70 18 DA 70 18 AF 32 82 60 3A 83 60 3C 32 83 60   += 061B
1860  FE 17 CA 70 18 DA 70 18 3E 10 32 83 60 CD 0C 06   += 060B
1870  C9 C3 66 F0 C3 6D F1 C3 CD F1 C3 85 F1 C3 E5 F1   += 0C56
1880  C3 FD F1 C3 9D F1 00 AF C9 00 00 C9 C3 61 F0 C3   += 0A1A
1890  66 F0 C3 9D F3 C3 E1 F2 C3 1C F3 C3 66 F0 00 00   += 0A2A
18A0  03 F7 67 F7 00 00 00 00 00 00 00 00 00 00 00 00   += 0258
18B0  00 C3 15 F2 C3 2D F2 C3 43 F2 C3 59 F2 C3 6F F2   += 09D6
18C0  C3 89 F2 C3 A3 F2 C3 D3 F2 C3 DA F2 C3 BD F2 C3   += 0CE2
18D0  B5 F1 3E EF 06 FF C9 31 67 F7 3E 80 D3 C8 CD 40   += 0996
18E0  F3 21 EC F0 CD E2 F0 CD 6D F1 E6 7F 4F CD 85 F1   += 0BB1
18F0  FE 31 20 1F 0E 01 CD 58 F3 38 DC 21 80 00 CB 61   += 0676
1900  20 03 21 00 FC E5 06 01 16 00 1E 01 CD 9D F3 E1   += 059F
1910  20 C5 E9 FE 32 20 0F 21 3E 8E 22 FC FF 21 D3 C8   += 07F3
1920  22 FE FF C3 FC FF FE 33 20 06 AF D3 C8 C3 00 20   += 0961
1930  FE 34 20 0E DB C0 AF D3 C8 31 02 F7 11 00 04 C3   += 0747
1940  03 20 FE 03 20 91 CD 6D F1 4F CD 85 F1 FE 03 20   += 07B3
1950  F5 18 84 7E B7 C8 4F CD 85 F1 23 18 F6 1A 46 4C   += 07FD
1960  4F 4D 4F 4E 20 34 2E 30 20 52 6F 6C 66 2D 44 69   += 0478
1970  65 74 65 72 20 4B 6C 65 69 6E 20 28 43 29 20 31   += 04C8
1980  39 38 36 0D 0A 0D 0A 31 20 3D 20 46 6C 6F 70 70   += 0384
1990  79 20 42 6F 6F 74 0D 0A 32 20 3D 20 47 6F 20 45   += 040E
19A0  30 30 30 30 68 0D 0A 33 20 3D 20 47 6F 20 42 61   += 0368
19B0  6E 6B 20 32 30 30 30 68 0D 0A 34 20 3D 20 5A 45   += 038A
19C0  41 54 20 73 74 61 72 74 0D 0A 43 54 52 4C 2D 43   += 049F
19D0  3D 54 65 73 74 6D 6F 64 65 0D 0A 00 00 00 AF D3   += 051B
19E0  C8 ED 73 6B F1 31 02 F7 CD F9 00 F5 3E 80 D3 C8   += 09C2
19F0  F1 ED 7B 6B F1 C9 AF D3 C8 ED 73 6B F1 31 02 F7   += 0AAE
1A00  CD 4B 03 F5 3E 80 D3 C8 F1 ED 7B 6B F1 C9 AF D3   += 0A69
1A10  C8 ED 73 6B F1 31 02 F7 CD BD 00 F5 3E 80 D3 C8   += 0986
1A20  F1 ED 7B 6B F1 C9 AF D3 C8 ED 73 6B F1 31 02 F7   += 0AAE
1A30  CD F9 01 F5 3E 80 D3 C8 F1 ED 7B 6B F1 C9 AF D3   += 0B15
1A40  C8 ED 73 6B F1 31 02 F7 CD 08 02 F5 3E 80 D3 C8   += 08D3
1A50  F1 ED 7B 6B F1 C9 AF D3 C8 ED 73 6B F1 31 02 F7   += 0AAE
1A60  CD 11 02 F5 3E 80 D3 C8 F1 ED 7B 6B F1 C9 AF D3   += 0A2E
1A70  C8 ED 73 6B F1 31 02 F7 CD 1F 02 F5 3E 80 D3 C8   += 08EA
```

zu Abb. 8.6.10

```
1A80  F1 ED 7B 6B F1 C9 AF D3 C8 ED 73 6B F1 31 02 F7   += 0AAE
1A90  CD BD 03 F5 3E 80 D3 C8 F1 ED 7B 6B F1 C9 AF D3   += 0ADB
1AA0  C8 ED 73 6B F1 31 02 F7 CD BA 05 3E 80 D3 C8 ED   += 0980
1AB0  7B 6B F1 C9 AF D3 C8 ED 73 6B F1 31 02 F7 CD 4E   += 09EB
1AC0  04 3E 80 D3 C8 ED 7B 6B F1 C9 AF D3 C8 ED 73 6B   += 09FF
1AD0  F1 31 02 F7 CD 5E 04 3E 80 D3 C8 ED 7B 6B F1 C9   += 0930
1AE0  AF D3 C8 ED 73 6B F1 31 02 F7 79 32 85 60 CD 02   += 088F
1AF0  04 3E 80 D3 C8 ED 7B 6B F1 C9 AF D3 C8 ED 73 6B   += 09FF
1B00  F1 31 02 F7 79 32 84 60 CD 02 04 3E 80 D3 C8 ED   += 07C3
1B10  7B 6B F1 C9 AF D3 C8 ED 73 6B F1 31 02 F7 79 32   += 097B
1B20  86 60 CD 02 04 3E 80 D3 C8 ED 7B 6B F1 C9 AF D3   += 0921
1B30  C8 ED 73 6B F1 31 02 F7 CD 03 00 3E 80 D3 C8 ED   += 08C4
1B40  7B 6B F1 C9 DB 70 E6 04 28 FA C9 CD D3 F2 79 D3   += 0A9E
1B50  70 C9 CD E7 F2 C3 9D F3 79 E6 03 FE 00 20 03 0E   += 08C3
1B60  11 C9 FE 01 20 03 0E 12 C9 FE 02 20 03 0E 91 C9   += 0570
1B70  0E 92 C9 E6 03 FE 00 20 03 0E 01 C9 FE 01 20 03   += 056D
1B80  0E 81 C9 FE 02 20 03 0E 02 C9 0E 82 C9 79 E6 7F   += 068B
1B90  4F F5 CD 02 F3 F1 E6 40 C2 31 F3 79 F6 30 4F C3   += 09B4
1BA0  35 F3 79 F6 20 4F 3E 02 C3 9E F3 F1 DB C0 FB ED   += 0A0E
1BB0  4D 3A 33 F0 CB 6F C2 4D F3 3E 00 C3 4F F3 3E 03   += 076A
1BC0  32 86 F5 3E FF 32 8D F5 C9 F3 3E D0 D3 C0 DB C0   += 0A96
1BD0  79 E6 8F 4F CD 3D F4 79 F6 30 4F CD 88 F3 D0 79   += 09BA
1BE0  E6 AF 4F CD 88 F3 D0 79 E6 8F F6 10 4F CD 88 F3   += 0A87
1BF0  D0 79 E6 8F 4F CD 88 F3 C9 79 D3 C4 3E 0F D3 C0   += 0A0E
1C00  DB C4 E6 40 28 FA DB C0 E6 98 37 C0 AF C9 AF 32   += 0A50
1C10  87 F5 ED 73 6B F1 31 02 F7 3E D0 D3 C0 3A 38 00   += 0875
1C20  32 88 F5 3E C3 32 38 00 E5 2A 39 00 22 89 F5 21   += 0623
1C30  3A F3 22 39 00 E1 DB C0 ED 56 FB CD 87 F4 F3 F5   += 0A72
1C40  3A 88 F5 32 38 00 E5 2A 89 F5 22 39 00 E1 F1 ED   += 07C8
1C50  7B 6B F1 C9 06 00 3A 8C F5 E6 80 28 11 3A 86 F5   += 07B5
1C60  E6 80 28 04 06 02 18 06 3A 87 F5 E6 02 47 DB C4   += 063C
1C70  E6 20 20 04 78 F6 04 C9 78 C9 79 D3 C4 7B D3 C2   += 08C6
1C80  0E C3 CD E3 F3 C6 88 D3 C0 DB C4 07 D2 18 F4 ED   += 0AC6
1C90  A2 C3 18 F4 79 D3 C4 7B D3 C2 0E C3 CD E3 F3 C6   += 0ACB
1CA0  A8 D3 C0 DB C4 07 D2 32 F4 ED A3 C3 32 F4 C5 01   += 0A18
1CB0  F4 01 CD 46 F4 C1 C9 0B 78 B1 C2 46 F4 C9 CD 3D   += 0989
1CC0  F4 79 CB AF D3 C4 7B D3 C2 7A D3 C3 06 18 3A 86   += 097C
1CD0  F5 E6 03 B0 D3 C0 CD 75 F4 79 D3 C4 06 1C 3A 86   += 0949
1CE0  F5 E6 03 B0 D3 C0 76 CD 3D F4 79 D3 C4 06 08 3A   += 08ED
1CF0  86 F5 E6 03 B0 D3 C0 76 78 B7 20 08 7A E6 83 32   += 0889
1D00  86 F5 AF C9 AF 32 8B F5 79 32 8C F5 E6 0F 4F 3A   += 08FE
1D10  8D F5 FE FF 28 14 E6 0F B9 CA E7 F4 E5 C5 21 8E   += 0A67
1D20  F5 4F 06 00 09 DB C1 77 C1 E1 E5 C5 3A 8C F5 E6   += 0953
1D30  0F 4F 06 00 21 8E F5 09 7E D3 C1 C1 E1 3A 8C F5   += 0780
1D40  32 8D F5 C5 01 FA 00 CD 46 F4 3A 8C F5 D3 C4 01   += 08CE
1D50  65 0B CD 46 F4 C1 18 06 DB C4 E6 20 20 0D 3A 8C   += 06EE
1D60  F5 4F CD 7C F5 DB C0 E6 80 20 F3 DB C1 FE FF 28   += 0B57
1D70  0F BA CA 2E F5 3A 8C F5 4F CD 7C F5 E6 90 28 1F   += 08BB
1D80  3A 8C F5 4F E5 D5 C5 CD 76 F4 C1 D1 E1 3A 8B F5   += 0AED
1D90  3C 32 8B F5 FE 0A 38 DD 3E FF B7 37 C9 AF C9 78   += 08EF
1DA0  FE 02 CA 5A F5 3A 8C F5 4F E5 D5 C5 CD 09 F4 C1   += 0A2D
1DB0  D1 E1 E6 9C 28 E7 E6 10 20 C6 3A 8B F5 3C 32 8B   += 08D2
```

zu Abb. 8.6.10

```
1DC0  F5 FE 05 38 DA FE 0A 38 B7 18 CD 3A 8C F5 4F E5     += 08D5
1DD0  D5 C5 CD 23 F4 C1 D1 E1 E6 FC 28 C1 E6 10 20 A0     += 0A72
1DE0  3A 8B F5 3C 32 8B F5 FE 04 38 95 18 AB E5 D5 C5     += 08B9
1DF0  CD 4D F4 C1 D1 E1 C9 00 00 00 00 00 00 00 FF FF     += 0748
1E00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00     += 0EF1
1E10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
1E90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     += 0000
```

1EA0..1FFF mit dem Wert 0 auffüllen

Abb. 8.6.10 Hexdump des Flomon 4.0

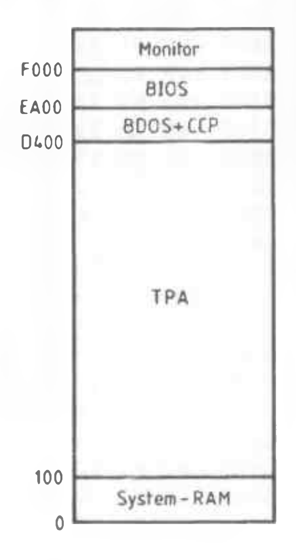


Abb. 8.6.11 Die Speicheraufteilung unter CP/;

8.6.2 Das BIOS für FLO-2 CP/M für den NDR-Klein-Computer

CP/M ist die Abkürzung für Control Program for Microcomputers, zu deutsch Steuerprogramm für Mikrocomputer. Das sagt noch nicht viel, denn das Grundprogramm des NDR-Klein-Computers steuert auch eine ganze Menge. Was CP/M so wertvoll macht, ist die Bedienung eines Floppy-Laufwerks. CP/M hat als eine seiner Hauptaufgaben, die Anordnung der Daten auf einer Diskette zu verwalten. CP/M ist also ein sogenanntes Dateiverwaltungssystem. Denn wenn man so viel Speicherplatz wie auf einer Diskette besitzt, muß man schnell und sicher auf die Daten zugreifen können. Dazu muß der Computer immer aktuell wissen, wo welche Daten und Programme mit welchem Namen auf der Diskette stehen. Das wird in dem Inhaltsverzeichnis einer jeden Diskette, dem Directory (oft Katalog genannt) mitnotiert. Dort sind die Namen aller Dateien verzeichnet, wie auch deren Blocknummern, woraus CP/M Spur und Sektor, also den physikalischen Ort der Datei und ihre Größe berechnen kann. Wenn zum Beispiel Daten oder Programme gelöscht werden sollen, so muß CP/M dafür sorgen, daß die Blöcke, in welchen der zu löschende Eintrag steht, wieder für neue Programme verfügbar werden. Man muß also Dateien anlegen können, man muß sie löschen können oder ein Inhaltsverzeichnis ausgeben können. All dies sind Aufgaben eines Betriebssystems, wie CP/M eines ist.

CP/M ist anpaßbar

CP/M hat aber auch noch eine weitere interessante Eigenschaft. Man kann es relativ einfach an die verschiedensten Gegebenheiten anpassen. Diese Eigenschaft war es, die CP/M so verbreitet hat. Weltweit gibt es wohl 10 Millionen Benutzer von CP/M-Rechnern. Deshalb gibt es für CP/M nahezu beliebig viel Software, von Programmiersprachen wie Pascal, Fortran, APL, C, Lisp, Algol, Forth oder Basic angefangen, über zahlreiche Programme aus der Textverarbeitung (Wordstar oder Kalkulationsprogramme wie Calcstar) bis zu Datenbankprogrammen oder höchstwertigen Spezial-Utilitys. Diese Anpaßfähigkeit war auch für uns ausschlaggebend. Denn CP/M läßt sich auch leicht an den NDR-Klein-Computer anpassen.

Zunächst noch etwas Allgemeines. *Abb. 8.6.11* zeigt die Speicheraufteilung unter CP/M. Von Adresse 0 bis 100h liegt der System-Speicher, den CP/M teilweise für eigene Zwecke nutzt. Ab 100h beginnt die sogenannte TPA (Transient Program Area). Sie nimmt die Anwenderprogramme auf. Ab Adresse D400h beginnt das eigentliche CP/M, wenn es so wie unseres angepaßt wird. Zunächst folgt ein Systemprogramm namens CCP (Consol Command Prozessor). Es hat die Aufgabe, Befehle von der Tastatur entgegenzunehmen und auszuwerten. Dann folgt das BDOS, was die eigentliche Diskettenverwaltung durchführt. Ab Adresse EA00 folgt das BIOS, das schließlich den anpaßbaren Teil, das System-Interface darstellt. Das BIOS muß man selbst schreiben, wenn man kein fertiges CP/M bezieht. Dieses BIOS ist in *Abb. 8.6.12* gezeigt, so wie man es für den NDR-Computer mit Z80-CPU gebrauchen kann. Das BIOS selbst greift wieder auf weitere Routinen zu, die bei uns ab Adresse F000h stehen und im EPROM mit FLOMON untergebracht sind. Dort stehen dann die eigentlichen Floppy-Unterprogramme, die den Zugriff auf die Diskette steuern. Diese Routinen erwarten Spur- und Sektor-Nummern, sie wollen nichts mehr von Dateinamen wissen. Wie gesagt, die Umsetzung der Namensangabe einer Datei, verbunden mit dem Schreib-, Lese- oder Lösch-Wunsch, in die Nummern der zu bearbeitenden Floppyscheibenbereiche, das ist die Aufgabe des BDOS. In das BIOS ist eingearbeitet, welche Laufwerke am Rechner angeschlossen sind. Da man an die FLO2 alle möglichen Laufwerke anschließen kann, würde das BIOS sehr umfangreich werden, wenn man alle Laufwerkparameter unterbringen wollte.

```
                           MACLIB DISKDEF  ;LOAD DEFINTION FOR DISKS

                    ;**************************************************
                    ; VERSION 3.1   VERSION 80 SPUR-LAUFWERK         *
                    ; GEBOOTET WIRD VOM 80 SPUR-LAUFWERK             *
                    ; A,B SIND DIE BEIDEN 80 SPUR LAUFWERKE          *
                    ; C,D IST EIN 8ZOLL LW, C=VORDERSEITE,D=RUECKS.  *
                    ; E IST RAM-FLOPPY                               *
                    ; (C) 1984 ROLF-DIETER KLEIN     841220          *
                    ;**************************************************


0016 =              VERS    EQU     22        ; DEFINITIONEN, ALLEGEMEIN
FFFF =              TRUE    EQU     OFFFFH
0000 =              FALSE   EQU     NOT TRUE
FFFF =              TEST    EQU     TRUE
```

zu Abb. 8.6.12

```
003C =          MSIZE   EQU 60          ; SPEICHERGROESSE, HIER 60K

A000 =          BIAS    EQU (MSIZE-20)*1024     ; MIN=20K
D400 =          CCP     EQU 3400H+BIAS  ; START DES CCP
DC06 =          BDOS    EQU CCP+806H    ; DORT BEGINNT DAS BDOS
EA00 =          BIOS    EQU CCP+1600H   ; UND DORT DAS BIOS
                ;
D400 =          CPMB    EQU CCP         ; START CP/M-BOOT.
                ;
1600 =          CPML    EQU BIOS-CPMB   ; LAENGE DES CP/MS
002C =          NSECTS  EQU CPML/128    ; ANZAHL DER BELEGTEN SEKTOREN
                ;
                ;
EA00                    ORG BIOS        ; START DES BIOS
                ;

0004 =          CDISK   EQU 4           ; ADRESSE IM SPEICHER, LETZTES LAUFWERK
0080 =          BUFF    EQU 80H         ; BUFFERADRESSE, DIE VOREINGESTELLT WIRD.
0005 =          RETRY   EQU 5           ; FEHLERVERSUCHE BEI BOOT ETC.

                ; VEKTORTABELLE DER BIOS-EINSPRUENGE

EA00 C3F2EA             JMP BOOT        ; KALT-START.
EA03 C314EB     WBOOTE: JMP WBOOT       ; WARM-START, BEI CTRL-C
EA06 C3E6EA             JMP CONST       ; CONSOL STATUS, ERGEBNIS IN A
EA09 C3E9EA             JMP CONIN       ; CONSOL EINGABE, NACH A
EA0C C3EFEA             JMP CONOUT      ; CONSOL AUSGABE VON C
EA0F C30FF0             JMP LIST        ; AUSGABE AUF DEN DRUCKER, VON C
EA12 C30CF0             JMP PUNCH       ; AUSGABE AUF PO, C-REGISTER
EA15 C306F0             JMP READER      ; EINGABE NACH RI, A-REGISTER

EA18 C3B2EB             JMP HOME        ; LAUFWERK NUR TRACK 0
EA1B C3B7EB             JMP SELDSK      ; LAUFWERK AUSWAEHLEN
EA1E C3CDEB             JMP SETTRK      ; SPUR AUSWAEHLEN
EA21 C3D2EB             JMP SETSEC      ; SEKTOR AUSWAEHLEN
EA24 C3EEEB             JMP SETDMA      ; ADRESSE FESTLEGEN
EA27 C301EC             JMP READ        ; SEKTOR LESEN
EA2A C377EC             JMP WRITE       ; SEKTOR SCHREIBEN
EA2D C3AFEB             JMP LISTST      ; DRUCKER FERTIG ?
EA30 C3D7EB             JMP SECTRAN     ; SEKTORUEBERSETZUNG
                ;
0004 =          OFFSET  EQU 4           ; FUER MINILW.

                ;
0185 =          DISKKAP EQU     389     ;TRUNC (5 * 1024 * (160-4)/2048) - 1
                ;                       ;CA. 780 K
                        DISKS 5         ; 5 LAUFWERKE HIERBEI
EA33+=          DPBASE  EQU     $       ;BASE OF DISK PARAMETER BLOCKS
EA33+00000000   DPE0:   DW      XLT0,0000H      ;TRANSLATE TABLE
EA37+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA3B+05EE83EA           DW      DIRBUF,DPB0     ;DIR BUFF,PARM BLOCK
EA3F+B6EE85EE           DW      CSV0,ALV0       ;CHECK; ALLOC VECTORS
EA43+00000000   DPE1:   DW      XLT1,0000H      ;TRANSLATE TABLE
EA47+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA4B+05EE83EA           DW      DIRBUF.DPB1     ;DIR BUFF,PARM BLOCK
EA4F+27EFF6EE           DW      CSV1,ALV1       ;CHECK, ALLOC VECTORS
EA53+A1EA0000   DPE2:   DW      XLT2,0000H      ;TRANSLATE TABLE
EA57+00000000           DW      0000H,0000H     ;SCRATCH AREA
EA5B+05EE92EA           DW      DIRBUF,DPB2     ;DIR BUFF,PARM BLOCK
EA5F+86EF67EF           DW      CSV2,ALV2       ;CHECK, ALLOC VECTORS
EA63+A1EA0000   DPE3:   DW      XLT3,0000H      ;TRANSLATE TABLE
```

zu Abb. 8.6.12

```
EA67+00000000            DW     0000H,0000H      ;SCRATCH AREA
EA6B+05EE92EA            DW     DIRBUF,DPB3      ;DIR BUFF,PARM BLOCK
EA6F+B5EF96EF            DW     CSV3,ALV3        ;CHECK, ALLOC VECTORS
EA73+00000000  DPE4:     DW     XLT4,0000H       ;TRANSLATE TABLE
EA77+00000000            DW     0000H,0000H      ;SCRATCH AREA
EA7B+05EEBBEA            DW     DIRBUF,DPB4      ;DIR BUFF,PARM BLOCK
EA7F+DCEFC5EF            DW     CSV4,ALV4        ;CHECK, ALLOC VECTORS
                         DISKDEF 0,0,39,0,2048,DISKKAP,256,256,OFFSET ; OFFSET=4
EA83+=         DPB0      EQU    $                ;DISK PARM BLOCK
EA83+2800                DW     40               ;SEC PER TRACK
EA85+04                  DB     4                ;BLOCK SHIFT
EA86+0F                  DB     15               ;BLOCK MASK
EA87+00                  DB     0                ;EXTNT MASK
EA88+8401                DW     388              ;DISK SIZE-1
EA8A+FF00                DW     255              ;DIRECTORY MAX
EA8C+F0                  DB     240              ;ALLOC0
EA8D+00                  DB     0                ;ALLOC1
EA8E+4000                DW     64               ;CHECK SIZE
EA90+0400                DW     4                ;OFFSET
0000+=         XLT0      EQU    0                ;NO XLATE TABLE
                         DISKDEF 1,0
EA83+=         DPB1      EQU    DPB0     ;EQUIVALENT PARAMETERS
0031+=         ALS1      EQU    ALS0     ;SAME ALLOCATION VECTOR SIZE
0040+=         CSS1      EQU    CSS0     ;SAME CHECKSUM VECTOR SIZE
0000+=         XLT1      EQU    XLT0     ;SAME TRANSLATE TABLE
                         DISKDEF 2,1,26,6,1024,243,64,64,2 ; 8 ZOLL DEFINITION
EA92+=         DPB2      EQU    $                ;DISK PARM BLOCK
EA92+1A00                DW     26               ;SEC PER TRACK
EA94+03                  DB     3                ;BLOCK SHIFT
EA95+07                  DB     7                ;BLOCK MASK
EA96+00                  DB     0                ;EXTNT MASK
EA97+F200                DW     242              ;DISK SIZE-1
EA99+3F00                DW     63               ;DIRECTORY MAX
EA9B+C0                  DB     192              ;ALLOC0
EA9C+00                  DB     0                ;ALLOC1
EA9D+1000                DW     16               ;CHECK SIZE
EA9F+0200                DW     2                ;OFFSET
EAA1+=         XLT2      EQU    $                ;TRANSLATE TABLE
EAA1+01                  DB     1
EAA2+07                  DB     7
EAA3+0D                  DB     13
EAA4+13                  DB     19
EAA5+19                  DB     25
EAA6+05                  DB     5
EAA7+0B                  DB     11
EAA8+11                  DB     17
EAA9+17                  DB     23
EAAA+03                  DB     3
EAAB+09                  DB     9
EAAC+0F                  DB     15
EAAD+15                  DB     21
EAAE+02                  DB     2
EAAF+08                  DB     8
EAB0+0E                  DB     14
EAB1+14                  DB     20
EAB2+1A                  DB     26
EAB3+06                  DB     6
EAB4+0C                  DB     12
EAB5+12                  DB     18
EAB6+18                  DB     24
EAB7+04                  DB     4
```

zu Abb. 8.6.12

```
EAB8+0A                  DB      10
EAB9+10                  DB      16
EABA+16                  DB      22
                         DISKDEF 3,2
EA92+=         DPB3      EQU     DPB2     ;EQUIVALENT PARAMETERS
001F+=         ALS3      EQU     ALS2     ;SAME ALLOCATION VECTOR SIZE
0010+=         CSS3      EQU     CSS2     ;SAME CHECKSUM VECTOR SIZE
EAA1+=         XLT3      EQU     XLT2     ;SAME TRANSLATE TABLE
                         DISKDEF 4,0,14,0,1024,180,64,64,0 ; RAM FLOPPY60K,120K,180K
EABB+=         DPB4      EQU     $                  ;DISK PARM BLOCK
EABB+0F00                DW      15                 ;SEC PER TRACK
EABD+03                  DB      3                  ;BLOCK SHIFT
EABE+07                  DB      7                  ;BLOCK MASK
EABF+00                  DB      0                  ;EXTNT MASK
EAC0+B300                DW      179                ;DISK SIZE-1
EAC2+3F00                DW      63                 ;DIRECTORY MAX
EAC4+C0                  DB      192                ;ALLOC0
EAC5+00                  DB      0                  ;ALLOC1
EAC6+1000                DW      16                 ;CHECK SIZE
EAC8+0000                DW      0                  ;OFFSET
0000+=         XLT4      EQU     0                  ;NO XLATE TABLE

               ; ENDEF AM SCHLUSS NICHT VERGESSEN

F01E =         MON80     EQU 0F01EH         ; NEUSTART DES MONITORS
F01E =         RMON80    EQU 0F01EH
000D =         CR        EQU 0DH            ; ZEICHENDEFINITIONEN
000A =         LF        EQU 0AH

               SIGNON:                      ; MELDUNG NACH DEM KALTSTART
EACA 1A                  DB 26
EACB 57656C636F          DB 'Welcome to 60 K CP/M 2.2'
EAE3 0D0A00              DB CR,LF,0

               CONST:                       ; CONSOL-STATUS
EAE6 C312F0              JMP 0F012H
               CONIN:
EAE9 CD03F0              CALL 0F003H        ; CONSOL-EINGABE
EAEC E67F                ANI 7FH            ; ACHTUNG PARITAET=0
EAEE C9                  RET
               CONOUT:                      ; CONSOL-AUSGABE
EAEF C309F0              JMP 0F009H

F00F =         LIST      EQU 0F00FH         ; DRUCKER
F00C =         PUNCH     EQU 0F00CH         ; PO
F006 =         READER    EQU 0F006H         ; RI

               BOOT:                        ; KALTSTART FOLGT HIER.
EAF2 310001              LXI SP,BUFF+80H ; STACK VORBELEGEN
EAF5 21CAEA              LXI H,SIGNON       ; MELDUNG AUSGEBEN
EAF8 CDF4EB              CALL PRMSG         ; MIT DRUCKROUTINE
EAFB AF                  XRA A              ; LAUFWERK A WIRD ANGEWAEHLT
EAFC 320400              STA CDISK          ;
               ;
               ; SEKTORENBUFFER IST LEER, MONITOR WIRD DESAKTIVIERT
               ;
EAFF AF                  XRA A              ; KEIN SCHREIBVORGANG MEHR AKTUELL
EB00 32F4ED              STA MWRTFLG        ; DAHER AUF 0 SETZEN
EB03 3EFF                MVI A,0FFH         ; LAUFWERK IST UNDEFINIERT
EB05 32F5ED              STA MDRVAKT        ; NACH DEM BOOTEN
EB08 2114EB              LXI H,WBOOT        ; MONITOREINSPRUNG WIRD
```

zu Abb. 8.6.12

```
EB0B 2234F0          SHLD 0F033H+1  ; KURZGESCHLOSSEN
EB0E 2237F0          SHLD 0F036H+1  ; DENN EVTL. UEBERSCHREIBEN


EB11 C36DEB          JMP GOCPM      ; UND CP/M DANN STARTEN

               WBOOT:               ; WARM-BOOT
EB14 3AF4ED          LDA MWRTFLG    ; WENN NOCH EIN ALTER TRACK ZUM
EB17 B7              ORA A          ; SCHREIBEN DA, DANN ZURUECK DAMIT.
EB18 CA1EEB          JZ NOTBAC      ; SONST WEITER.
EB1B CDC9ED          CALL PUTTRK    ; NORMALERWEISE IST SCHREIBVORGANG
                                    ; NACH EINEM DIREKTORYZUGRIFF ABGESCHLOSSEN
                                    ;
EB1E 3EFF      NOTBAC: MVI A,0FFH   ; ALLE TRACKS UNGUELTIG, BEI DISKETTENWECHSEL
EB20 32F5ED          STA MDRVAKT    ; WICHTIG.

EB23 318000          LXI SP,BUFF    ; STACK ZUWEISEN
EB26 0E05            MVI C,RETRY    ; ANZAHL DER VERSUCHE
EB28 C5              PUSH B         ; UND DANN ANFANGEN ZU BOOTEN
EB29 0100D4    WBOOT0: LXI B,CPMB   ; BOOT VON MINI-DISKETTE
EB2C CDEEEB          CALL SETDMA    ; AUF DER STARTADRESSE DES CP/MS
EB2F 0E00            MVI C,0        ; LAUFWERK A
EB31 CD87EB          CALL SELDSK    ; AUSWAEHLEN
EB34 0E00            MVI C,0        ; TRACK 0
EB36 CDCDEB          CALL SETTRK    ; UND DEN ZWEITEN PHYS. SEKTOR, (NR8 LOGISCH)
EB39 0E08            MVI C,8        ; ENSTPRICHT NR 2 BEI 1024 BYTES
EB3B CDD2EB          CALL SETSEC    ; WICHTIG, DA ANDERE ZAEHLWEISE
EB3E C1              POP B          ; UND VON DA AN N SEKTOREN EINLESEN
EB3F 062C            MVI B,NSECTS   ; ABER DAS BIOS NICHT UEBERSCHREIBEN
                                    ; DAMIT PATCHES LEICHT MOEGLICH SIND
EB41 C5        RDSEC: PUSH B        ; ANZAHL MERKEN
EB42 CD01EC          CALL READ      ; LESEN AUSFUEHREN
EB45 C297EB          JNZ BOOTERR    ; FEHLER: DEFEKTER SEKTOR
EB48 2AFEED          LHLD IOD       ; ZIELADRESSE LADEN
EB4B 118000          LXI D,128      ; UM LOGISCHE SEKTORGROESSER ERHOEHEN
EB4E 19              DAD D          ; DAZU ADDIEREN,
EB4F 22FEED          SHLD IOD       ; UND DANN WIEDER ZURUECKSPEICHERN.
EB52 3AFDED          LDA IOS        ; SEKTOR LADEN
EB55 FE27            CPI 39         ; MINI 0..39 SEKTOREN A 128 BYTES
EB57 DA64EB          JC RD1         ; SOLANGE AUF DER GLEICHEN SPUR BLEIBEN
EB5A 3AFCED          LDA IOT        ; DANN NEUE SPUR ANWAEHLEN,
EB5D 3C              INR A          ; JEDOCH IM VERFAHREN 0,2,...
EB5E 3C              INR A          ; SPUR 0, DANN SPUR 2, WEGEN BOOT.ASM
EB5F 32FCED          STA IOT        ; DENN 1,3,5 IST DIE RUECKSEITE DES LAUFWERKS
EB62 3EFF            MVI A,0FFH     ; 0,1,2,3.... NACH INCREMENT A=0
EB64 3C        RD1:  INR A          ; DANN 0
EB65 32FDED          STA IOS        ; UND AUCH NEUESN SEKTOR ANWAEHLEN
EB68 C1              POP B          ; SCHLEIFENZAEHLER ZURUECH
EB69 05              DCR B          ; UND IMMER WEITER LESEN
EB6A C241EB          JNZ RDSEC      ; DANACH CP/M NEU STARTEN
               GOCPM:
EB6D 018000          LXI B,BUFF     ; BUFFER AUF DEFAULT EINSTELLEN
EB70 CDEEEB          CALL SETDMA    ; SO WIE ES CP/M BRAUCHT
EB73 3EC3            MVI A,JMP      ; SPRUNG AUF DEN WARM-BOOT IM
EB75 320000          STA 0          ; RAM ABLEGEN.
EB78 2103EA          LXI H,WBOOTE   ; WARM-BOOT-ADRESSE
EB7B 220100          SHLD 1         ; NICHT VERGESSEN
EB7E 320500          STA 5          ; SPRUNG AUF DIE BDOS-CALL-ADRESSE
EB81 2106DC          LXI H,BDOS     ; LEGEN UND AUCH DAS ZIEL
EB84 220600          SHLD 6         ; DORTHIN
EB87 323800          STA 7*8        ; RST7 DEFINIEREN, DEFAULT IST MONITOR
```

zu Abb. 8.6.12

```
EBBA 211EF0          LXI H,MONBO     ; DER ABER NORMALERWEISE KURZGESCHL. IST.
EBBD 223900          SHLD 7*8+1
EB90 3A0400          LDA CDISK       ; DAS ZULETZT VERWENDETE LAUFWERK
EB93 4F              MOV C,A         ; LADEN UND DAMIT SELEKTIEREN.
EB94 C300D4          JMP CPMB

          BOOTERR:                   ; IM FEHLERFALLE, BEI BAD-SEKTOR.
EB97 C1              POP B           ; ERST MAL NOCHEINMAL VERSUCHEN
EB98 0D              DCR C           ;
EB99 CAA0EB          JZ BOOTERO      ; BIS HOFFNUNGSLOS, DANN FEHLERMELDUNG
EB9C C5              PUSH B
EB9D C329EB          JMP WBOOTO      ; TRY AGAIN

          BOOTERO:                   ; FEHLERMELDUNG SCHLIESSLICH AUSGEBEN
EBA0 21A9EB          LXI H,BOOTMSG   ; UND MONITOR NEU STARTEN, BZW. WBOOT.
EBA3 CDF4EB          CALL PRMSG
EBA6 C31EF0          JMP MONBO

          BOOTMSG:                   ; FEHLERMELDUNG
EBA9 3F424F4F54      DB '?BOOT',0

          LISTST:                    ; LST-STATUS, DERZEIT KURZGESCHLOSSEN
EBAF 00              NOP             ;
EBB0 AF              XRA A           ; GGF. HIER SPRUNG EINBAUEN.
EBB1 C9              RET

          HOME:                      ; LAUFWERK, SPUR 0 ANFAHREN
EBB2 0E00            MVI C,0         ; ABER NUR ANWAEHLEN, NICHT
EBB4 C3CDEB          JMP SETTRK      ; AUSFUEHREN

          SELDSK:                    ; LAUFWERK AUSWAEHLEN
EBB7 210000          LXI H,0         ; UND PRUEFEN OB GUELTIG
EBBA 79              MOV A,C         ; WENN GROESSER ALS NDISKS
EBBB FE05            CPI NDISKS      ; DANN NICHT OK
EBBD D0              RNC             ; NUMMER 0 BIS N-1 ERSCHEINT IN A
EBBE 32F9ED          STA DBANK       ; DANN ZUSAETZLICH DIE
EBC1 69              MOV L,C         ; LAUFWERKSTABELLE AUSRECHNEN
EBC2 2600            MVI H,0         ; DAZU NUMMER MIT 16 MULTIPLIZIEREN
EBC4 29              DAD H           ;
EBC5 29              DAD H           ;
EBC6 29              DAD H           ;
EBC7 29              DAD H           ;
EBC8 1133EA          LXI D,DPBASE    ; UND BASISADRESSE DRAUF ADDIEREN.
EBCB 19              DAD D
EBCC C9              RET

          SETTRK:                    ; SPUR MERKEN
EBCD 21FCED          LXI H,IOT       ; DAZU IM SPEICHERZELLE LADEN
EBD0 71              MOV M,C
EBD1 C9              RET

          SETSEC:                    ; SEKTOR MERKEN
EBD2 21FDED          LXI H,IOS       ; DAZU IN SPEICHERZELLE LADEN
EBD5 71              MOV M,C
EBD6 C9              RET

          SECTRAN:                   ; SEKTOR UMSETZUNG
EBD7 7A              MOV A,D         ; WENN EIN SKEW-FAKTOR VERWENDET WIRD,
EBD8 B3              ORA E           ; WIE Z.B. BEI 8 ZOLL UEBLICH.
EBD9 CAE6EB          JZ SE1          ; =0, DANN KEIN SKEW VERWENDET,
```

zu Abb. 8.6.12

```
EBDC 0600            MVI B,0         ; SONST IN DE ADRESSE DER SKEW-TABELLE
EBDE EB              XCHG            ; DAZU SEKTOR IN C ADDIEREN
EBDF 09              DAD B           ; UND WERT ALS NEUEN SEKTOR
EBE0 7E              MOV A,M         ; FESTLEGEN UND
EBE1 32FDED          STA IOS         ; SPEICHERN.
EBE4 6F              MOV L,A
EBE5 C9              RET
EBE6 69     SE1:     MOV L,C         ; SONST NUR EINFACHE WERT
EBE7 79              MOV A,C         ; UEBERNEHMEN, OHNE UMRECHNUNG.
EBE8 32FDED          STA IOS         ; AUCH MERKEN
EBEB 2600            MVI H,0         ; SE 0..255 MAX
EBED C9              RET
            ;

            SETDMA:                  ; ADRESSE FUER FLOPPY-ZUGRIFF FESTLEGEN.
EBEE 69              MOV L,C
EBEF 60              MOV H,B
EBF0 22FEED          SHLD IOD
EBF3 C9              RET

            PRMSG:                   ; TEXT AUSGEBEN, FUER FEHLERMELDUNG
EBF4 7E              MOV A,M         ; DAZU LADEN
EBF5 B7              ORA A           ; =0, DANN ENDE DES TEXTES
EBF6 C8              RZ              ; SONST UEBER CONSOLE AUSGEBEN
EBF7 E5              PUSH H
EBF8 4F              MOV C,A
EBF9 CDEFEA          CALL CONOUT
EBFC E1              POP H
EBFD 23              INX H           ; BIS ALLE BUCHSTABEN DRAUSSEN
EBFE C3F4EB          JMP PRMSG

            ;READ UND WRITE UNTER VERWENDUNG VON EXEC IM MONITOR
            ; HL=DMA ADR
            ; DE=TRACK/SECTOR
            ; B=0 RSTORE
            ;   1 READ
            ;   2 WRITE
            ; C=DRIVE 0...3          ; BEI MEXEC, EXEC
            ; BEI FEXEC IST C BEI BESTIMMT.

            READ:                    ; EINEN SEKTOR LESEN
EC01 3AF9ED          LDA DBANK       ; DAZU LAUFWERK BESTIMMEN
EC04 FE02            CPI 2           ;
EC06 DADCEC          JC MINIRD       ; 0,1 SIND MINILAUFWERKE
EC09 FE04            CPI 4
EC0B DA57EC          JC MAXIREAD     ; 8 ZOLL LAUFWERK
            ;
            ; RAMFLOPPY ZUSATZ-ROUTINEN
            ;
EC0E CD23EC          CALL ADRERZ     ; HL=QUELLADRESSE
EC11 EB              XCHG            ; ADRESSUMRECHNUNG DURCHFUEHREN
EC12 2AFEED          LHLD IOD        ; ZIELADRESSE LADEN
EC15 EB              XCHG            ; UND DE=ZIEL, HL=QUELLE, C=BANK QUELLE
EC16 0600            MVI B,0         ; ZIEL IST BANK 0
EC18 CDF1ED          CALL REXEC      ; UND 128 BYTES KOPIEREN, CARRY=FEHLER
EC1B D221EC          JNC NORERR
EC1E 3E01            MVI A,1         ; FEHLER DA, BANK NICHT VORHANDEN,
EC20 C9              RET             ; WIRKT WIE BAD-SEKTOR
EC21 AF     NORERR:  XRA A           ; SONST OK.
EC22 C9              RET
```

zu Abb. 8.6.12

```
                        ; SEKTOR 0..E,  TRACK 0..5FH
                        ; SSSSTTTT T0000000   , ADRESSE FUER RAM-FLOPPY
                        ADRERZ:                    ; ADRESSE BERECHNEN, QUELLE IN HL
EC23 3AFCED                     LDA IOT            ; TRACK HOLEN
EC26 0F                         RRC                ; UND UMRECHNEN
EC27 E60F                       ANI 0FH            ; UNTERER TEIL VOM MSB
EC29 67                         MOV H,A
EC2A 3AFDED                     LDA IOS            ; DANN SEKTOR DAZU
EC2D 07                         RLC
EC2E 07                         RLC
EC2F 07                         RLC
EC30 07                         RLC
EC31 E6F0                       ANI 0F0H
EC33 B4                         ORA H
EC34 67                         MOV H,A            ; DAMIT SSSSTTTT OK
EC35 3AFCED                     LDA IOT
EC38 0F                         RRC                ; TXXXXXXX
EC39 E680                       ANI 80H
EC3B 6F                         MOV L,A            ; T0000000
EC3C 3AFCED                     LDA IOT            ; NUN NOCH BANK BESTIMMEN, IN C UND B
EC3F 07                         RLC                ; UND DAZU MSB-TEIL DES TRACKS VERWENDEN
EC40 07                         RLC                ; OMMTTTTT
EC41 07                         RLC                ; BANKNUMMER
EC42 E603                       ANI 03H            ; 000000MM
EC44 C601                       ADI 1              ; ERST AB BANK 1 STARTEN
EC46 47                         MOV B,A            ; DA BANK 0=CP/M RAM UND TPA
EC47 4F                         MOV C,A            ; OK BEIDE DEFINIERT
EC48 C9                         RET

                        NEUBANK:                   ; UMRECHNEN FUER 8ZOLL
                                                   ; UND NEUEN FLOPPY-EINSPRUNG VERWENDEN.
                                                   ; NACH C LADEN
                                                   ; LW=2, DANN VORDERSEITE LW 3
                                                   ; LW=3, DANN RUECKSEITE LW 3
EC49 3AF9ED                     LDA DBANK          ;
EC4C FE02                       CPI 2
EC4E C254EC                     JNZ NEU1
EC51 0E14                       MVI C,00010100B ; SD,8 ZOLL, LW=3
EC53 C9                         RET
EC54 0E94               NEU1:   MVI C,10010100B
EC56 C9                         RET

                        MAXIREAD:                  ; LESEN DER 8 ZOLL FLOPPY
EC57 0605               SK1:    MVI B,RETRY        ; ANZAHL DER LESEVERSUCHE
EC59 C5                 LP:     PUSH B             ; DANN AUSFUEHREN
EC5A 2AFEED                     LHLD IOD           ; ZIELADRESSE HOLEN
EC5D 3AFCED                     LDA IOT            ; SPURNUMMER
EC60 57                         MOV D,A
EC61 3AFDED                     LDA IOS            ; SEKTORNUMMER
EC64 5F                         MOV E,A
EC65 0601                       MVI B,1            ; LESE-BEFEHLSCODE
EC67 CD49EC                     CALL NEUBANK       ; VORHER LAUFWERKSCODE UMRECHNEN
EC6A CDE8ED                     CALL FEXEC         ; UND DANN AUSFUEHREN
EC6D C1                         POP B              ; RETRY-ZAEHLER
EC6E C8                         RZ                 ; KEIN FEHLER, DANN OK ZURUECK
EC6F 05                         DCR B              ; SONST NOCHMALS PROBIEREN
EC70 C259EC                     JNZ LP
EC73 3E01                       MVI A,1            ; BAD SEKTOR
EC75 B7                         ORA A
EC76 C9                         RET
```

zu Abb. 8.6.12

```
                WRITE:                          ; SCHREIBEN EINES SEKTORS
EC77 3AF9ED             LDA DBANK       ; DAZU LAUFWERKSCODE LADEN
EC7A FE02               CPI 2           ; UND FLOPPY-TYP BESTIMMEN
EC7C DA35ED             JC MINIWR       ; 0 UND 1 SIND MINILAUFWERKE
EC7F FE04               CPI 4           ; 2 UND 3 MAXILAUFWERKE
EC81 DA96EC             JC MAXIWR       ; REST IST RAM-FLOPPY
                ; RAM FLOPPY
EC84 CD23EC             CALL ADRERZ     ; HL=QUELLADRESSE
EC87 EB                 XCHG            ; UMRECHNEN
EC88 2AFEED             LHLD IOD        ; ZIEL IN BANK HL=QUELLE DIESMAL
EC8B 0E00               MVI C,0         ; QUELLE IST BANK 0, B=ZIEL, DE=ZIEL
EC8D CDF1ED             CALL REXEC      ; UND 128 BYTES KOPIEREN, CARRY=FEHLER
EC90 D221EC             JNC NORERR      ; OK BANK WAR DA, SONST
EC93 3E01               MVI A,1         ; FEHLER AUSGEBEN
EC95 C9                 RET

EC96 0605       MAXIWR: MVI B,RETRY     ; SCHREIBEN BEI 8 ZOLL
EC98 C5         LPP:    PUSH B          ; DAZU FEHLERZAEHLER RETTEN
EC99 2AFEED             LHLD IOD        ; QUELLEADRESSE LADEN
EC9C 3AFCED             LDA IOT         ; SPUR
EC9F 57                 MOV D,A
ECA0 3AFDED             LDA IOS         ; SEKTOR
ECA3 5F                 MOV E,A
ECA4 0602               MVI B,2         ; SCHREIB-BEFEHLSCODE
ECA6 CD49EC             CALL NEUBANK    ; VORHER LAUFWERK UMRECHNEN
ECA9 CDE8ED             CALL FEXEC      ; UND AUSFUEHREN
ECAC C1                 POP B
ECAD C8                 RZ              ; KEIN FEHLER, DANN ZURUECK
ECAE 05                 DCR B           ; SONST ERNEUT VERSUCHEN
ECAF C298EC             JNZ LPP
ECB2 3E01               MVI A,1         ; BAD SEKTOR
ECB4 B7                 ORA A
ECB5 C9                 RET

            ; MINIFLOPPY 80 SPUR, DD, DS
            ;
            ;READ UND WRITE UNTER VERWENDUNG VON MEXEC
            ; HL=DMA ADR
            ; DE=TRACK/SECTOR
            ; B=0 RSTORE
            ;   1 READ
            ;   2 WRITE
            ; C=DRIVE 0...3  10H,11H,12H,13H  DOUBMIN 0D0H,0D1H,0D2H,0D3H
            ;                                          A    C    B    D
            ; 1K BUFFER IN MONITORGEBIET
            ; WIRD DADURCH TEILWEISE UEBERSCHRIEBEN
            ; BEI WARMBOOT MUSS BUFFER GELEERT WERDEN
            ; DEBLOCK WIRD AUS SICHERHEITSGRUENDEN NICHT VERWENDET
            ;

FC00 =      BUFFER  EQU     0FC00H  ; FREIES GEBIET BIS FFFF NUR MONITORBEFEHLE

            CALC:.                  ;RECHNET DBANK IN PHYS LAUFWERK UM
                                    ;RECHNET IOS IN SEKTORBUFFERNR UM
ECB6 3AFDED         LDA IOS         ;0..39 IST DER BEREICH
ECB9 0F             RRC             ;X000NNNN 0..15
ECBA 0F             RRC
ECBB 0F             RRC             ;0,1,2,3,4
ECBC E607           ANI 00000111B   ; MAX
ECBE 3C             INR A           ;1,2,3,4,5  STARTSEKTOR DES GEBIETS (1K)
ECBF 5F             MOV E,A         ;IN E ALS PARAMETER
```

zu Abb. 8.6.12

```
ECC0 3AF9ED          LDA DBANK       ;DRIVE 0->0  1->2
             ;       -- NUR 0,1 SUI 4 ;UND TRACK UMRECHNEN , LAUFWERK 0,1
ECC3 FE01            CPI 1           ; 0,2,4,6,8 IST VORDERSEITE 1,3,5...RUECKSEITE
ECC5 C2CAEC          JNZ CAL2
ECC8 3E02            MVI A,2
ECCA 4F      CAL2:   MOV C,A         ;DRIVE PHYSIKALISCH
ECCB 3AFCED          LDA IOT
ECCE 0F              RRC             ; TRACK / 2,  = PHYS TRACK, CARRY=RUECKSEITE
ECCF 57              MOV D,A         ; TRACK MERKEN
ECD0 D2D7EC          JNC CAL3
ECD3 79              MOV A,C
ECD4 F601            ORI 1           ; D0 -> D1,  D2 -> D3
ECD6 4F              MOV C,A
ECD7 7A      CAL3:   MOV A,D
ECD8 E67F            ANI 7FH         ; BEREICH TRACK 0..79 REAL
ECDA 57              MOV D,A         ;TRACK=D SEKTOR=E DRIVE=C
ECDB C9              RET

             ;
ECDC CDB6EC  MINIRD: CALL CALC       ; FUER VERGLEICH, LAUFWERKSDATEN UMRECHNEN
ECDF 3AF5ED          LDA MDRVAKT     ; UND NUN AKTUELLES LAUFWERK VERGLEICHEN
ECE2 B9              CMP C           ; WENN NICHT GLEICH, DANN NEU
ECE3 C210ED          JNZ RLOAD       ; LADEN,
ECE6 3AF6ED          LDA MTRKAKT     ; SONST SPUR VERGLEICHEN
ECE9 BA              CMP D           ; UND
ECEA C210ED          JNZ RLOAD
ECED 3AF7ED          LDA MSEKAKT     ; SONST SEKTOR
ECF0 BB              CMP E           ; WENN GLEICH, DANN SEKTOR IM SPEICHER
ECF1 C210ED          JNZ RLOAD       ; UND LADEN UNNOETIG
             R1RD:                   ; OK IST SCHON IN BUFFER
ECF4 2100FC          LXI H,BUFFER    ; ADRESSE BERECHNEN
ECF7 3AFDED          LDA IOS         ; 0..39   * 128 + BUFFER
ECFA E607            ANI 00000111B   ; 0,1,2,3,4,5,6,7
ECFC 57                 MOV D,A
ECFD 1E00               MVI E,0        ; SCHIEBEN MIT Z80 BEFS
ECFF CB2A               DB 0CBH,2AH    ; SRA D
ED01 CB1B               DB 0CBH,1BH    ; RR E   = *256/2
ED03 19                 DAD D          ; +BUFFER
ED04 EB                 XCHG           ; NACH DE IST ZIEL
ED05 2AFEED             LHLD IOD       ; DMA ADRESSE QUELLE HIER
ED08 EB                 XCHG           ; ZIEL DE
ED09 018000             LXI B,128      ; LAENGE
ED0C EDB0               DB 0EDH,0B0H   ; LDIR
ED0E AF                 XRA A
ED0F C9                 RET            ; OK ENDE

              RLOAD:                   ; NEUEN LADEN, GGF ALTEN ZURUECKSCHREIBEN
ED10 3AF4ED             LDA MWRTFLG
ED13 B7                 ORA A
ED14 CA1DED             JZ R1LOAD
ED17 CDC9ED             CALL PUTTRK    ; ALTEN ZURUECKSCHREIBEN
ED1A DAA4ED             JC ERRIO
ED1D CDB6EC   R1LOAD:   CALL CALC      ; BERECHNEN
ED20 79                 MOV A,C
ED21 32F5ED             STA MDRVAKT
ED24 7B                 MOV A,E
ED25 32F7ED             STA MSEKAKT
ED28 7A                 MOV A,D
ED29 32F6ED             STA MTRKAKT
ED2C CDA8ED             CALL GETTRK    ; NEUEN SEKTOR LESEN
ED2F DAA4ED             JC ERRIO       ; FEHLER AUFGETRETEN
```

zu Abb. 8.6.12

```
ED32 C3F4EC          JMP R1RD       ; ENDE
               ;

               MINIWR:              ; SCHREIBEN EINES SEKTORS
ED35 79              MOV A,C
ED36 32F8ED          STA ALLOC      ; INFORMATION 1=DIREKTORY WRITE
ED39 CDB6EC          CALL CALC      ; FUER VERGLEICH, BERECHUNG AUSFUEHREN
ED3C 3AF5ED          LDA MDRVAKT    ; UND WIE BEI MINIRD
ED3F B9              CMP C          ;
ED40 C27FED          JNZ WLOAD
ED43 3AF6ED          LDA MTRKAKT
ED46 BA              CMP D
ED47 C27FED          JNZ WLOAD
ED4A 3AF7ED          LDA MSEKAKT
ED4D BB              CMP E
ED4E C27FED          JNZ WLOAD      ; LADEN UNNOETIG, SEKTOR SCHON DA.
               W1WR:                ; OK IST SCHON IN BUFFER
ED51 2100FC          LXI H,BUFFER   ; ADRESSE BERECHNEN
ED54 3AFDED          LDA IOS        ; 0..39   * 128 + BUFFER
ED57 E607            ANI 00000111B  ; 0,1,2,3,4,5,6,7
ED59 57              MOV D,A
ED5A 1E00            MVI E,0        ; SCHIEBEN MIT Z80 BEFS
ED5C CB2A            DB 0CBH,2AH    ; SRA D
ED5E CB1B            DB 0CBH,1BH    ; RR E   = *256/2
ED60 19              DAD D          ; +BUFFER
ED61 EB              XCHG           ; NACH DE IST ZIEL
ED62 2AFEED          LHLD IOD       ; DMA ADRESSE QUELLE HIER
ED65 018000          LXI B,128      ; LAENGE ZIEL DE
ED68 EDB0            DB 0EDH,0B0H   ; LDIR
ED6A 3E01            MVI A,1
ED6C 32F4ED          STA MWRTFLG    ; NUN BESCHRIEBEN
ED6F 3AF8ED          LDA ALLOC      ; =1 DANN ZURUECK
ED72 FE01            CPI 1           ; WENN DIREKTORY ZUGRIFF, DANN
ED74 C27DED          JNZ W2WR        ; GLEICH ZURUECKSCHREIBEN.
ED77 CDC9ED          CALL PUTTRK     ;
ED7A DAA4ED          JC ERRIO        ; FALLS FEHLER, DANN BAD SEKTOR
               W2WR:
ED7D AF              XRA A           ; KEIN FEHLER
ED7E C9              RET             ; OK ENDE

               WLOAD:                ; NEUEN LADEN, GGF ALTEN ZURÜECKSCHREIBEN
ED7F 3AF4ED          LDA MWRTFLG     ;
ED82 B7              ORA A
ED83 CA8CED          JZ W1LOAD
ED86 CDC9ED          CALL PUTTRK     ; ALTEN ZURUECKSCHREIBEN
ED89 DAA4ED          JC ERRIO
ED8C CDB6EC    W1LOAD: CALL CALC     ; BERECHNEN
ED8F 79              MOV A,C
ED90 32F5ED          STA MDRVAKT
ED93 7B              MOV A,E
ED94 32F7ED          STA MSEKAKT
ED97 7A              MOV A,D
ED98 32F6ED          STA MTRKAKT
ED9B CDA8ED          CALL GETTRK
ED9E DAA4ED          JC ERRIO
EDA1 C351ED          JMP W1WR

EDA4 3E01      ERRIO: MVI A,1        ; FEHLER AUFGETRETEN
EDA6 B7              ORA A
EDA7 C9              RET

               ; BUFFERVERWALTUNG
```

zu Abb. 8.6.12

```
                GETTRK:                         ;TRKAKT,SEKAKT,DRVAKT ENTHALTEN NEUE
                                                ;BUFFERADRESSE
EDA8 AF                 XRA A
EDA9 32F4ED             STA MWRTFLG             ; SEKTOR EINLESEN
EDAC 2100FC             LXI H,BUFFER
EDAF 3AF7ED             LDA MSEKAKT
EDB2 5F                 MOV E,A
EDB3 3AF5ED             LDA MDRVAKT
EDB6 F6D0               ORI 11010000B           ;LAUFWERK PHYS 0,1,2,3 DOUBLE DENSE
EDB8 4F                 MOV C,A
EDB9 0601               MVI B,1                 ;READ, 1K DIREKT
EDBB 3AF6ED             LDA MTRKAKT
EDBE 57                 MOV D,A
EDBF CDEEED             CALL MEXEC
EDC2 DAC7ED             JC ERRX
EDC5 AF                 XRA A
EDC6 C9                 RET
                ;
EDC7 37         ERRX:   STC
EDC8 C9                 RET


                PUTTRK:                         ;TRKAKT,SEKAKT,DRVAKT ENTHALTEN NEUE
                                                ;BUFFERADRESSE
EDC9 AF                 XRA A                   ;WIRD ZURUECKGESCHRIEBEN
EDCA 32F4ED             STA MWRTFLG
EDCD 2100FC             LXI H,BUFFER
EDD0 3AF7ED             LDA MSEKAKT
EDD3 5F                 MOV E,A
EDD4 3AF5ED             LDA MDRVAKT
EDD7 F6D0               ORI 11010000B           ;LAUFWERK PHYS 0,1,2,3 DOUBLE DENSE
EDD9 4F                 MOV C,A
EDDA 0602               MVI B,2                 ;WRITE  1K SEKTOR
EDDC 3AF6ED             LDA MTRKAKT
EDDF 57                 MOV D,A
EDE0 CDEEED             CALL MEXEC
EDE3 DAC7ED             JC ERRX
EDE6 AF                 XRA A
EDE7 C9                 RET

EDE8 C321F0     FEXEC:  JMP 0F021H              ;FLOMON NEUER VEKTOR
EDEB C324F0     EXEC:   JMP 0F024H              ;FLOMON UND MC
EDEE C327F0     MEXEC:  JMP 0F027H              ;FLOMON UND MC
EDF1 C35BF0     REXEC:  JMP 0F05BH              ;FLOMON RAM-FLOPPY


                                                ;SOFT SYSTEM
                ;
                : RAM ZELLEN

EDF4 00         MWRTFLG:        DB 0            ;<>0 IST WRITEN
EDF5 00         MDRVAKT:        DB 0            ;DRIVE DAS GELADEN IST
EDF6 00         MTRKAKT:        DB 0            ;TRACK
EDF7 00         MSEKAKT:        DB 0            ;SEKTOR
EDF8 00         ALLOC:          DB 0            ; MERKER, 1=DIREKTORY WRITE
                ;


                ;
EDF9 00         DBANK:  DB 0
EDFA 80         IOPB:   DB 80H  ;NORM IO
EDFB 01         ION:    DB 1    ;SECTOR NR
```

zu Abb. 8.6.12

```
EDFC 04         IOT:    DB OFFSET ;TRK
EDFD 01         IOS:    DB 1
EDFE 8000       IOD:    DW BUFF
EE00 01         ALTDRV: DB 1
EE01 0000       INDADR: DW 0
EE03 0000       INDADR2:DW 0


                        ENDEF
EE05+=          BEGDAT  EQU     $
EE05+           DIRBUF: DS      128     ;DIRECTORY ACCESS BUFFER
EE85+           ALV0:   DS      49
EEB6+           CSV0:   DS      64
EEF6+           ALV1:   DS      49
EF27+           CSV1:   DS      64
EF67+           ALV2:   DS      31
EF86+           CSV2:   DS      16
EF96+           ALV3:   DS      31
EFB5+           CSV3:   DS      16
EFC5+           ALV4:   DS      23
EFDC+           CSV4:   DS      16
EFEC+=          ENDDAT  EQU     $
01E7+=          DATSIZ  EQU     $-BEGDAT

EFEC                    END
```

Abb. 8.6.12 Das ist das BIOS für den NDR-Computer mit Z80

Das neue BIOS

Mit dem neuen BIOS kann man zunächst 5¼-Zoll-Laufwerke mit 80 Spuren, zwei Seiten und doppelter Aufzeichnungsdichte ansteuern. Man erhält dabei eine Speicherkapazität von etwa 780 KByte (abzüglich Inhaltsverzeichnis). CP/M selbst ist auf den äußeren Spuren der Diskette untergebracht. Es sind dafür 20 KByte reserviert, wobei das CP/M-2.2 allein nicht soviel benötigen würde. Da aber CP/M-68K für den 68008 und 68000 so viel Platz benötigt und Sie Ihren Computer ja einmal auf 16 Bit ausbauen wollen, seien etwa 10 KByte Systemplatz verschenkt. Sie können dann die Z80-Disketten auch mit dem 68000 vollständig lesen und geeignet bearbeiten. Insgesamt hat man mit unserem neuen Format wirklich ein Optimum an Kapazität erreicht.

Ein paar Besonderheiten des BIOS sollen noch erwähnt werden. Zunächst einmal sind die Laufwerke A und B mit 80 Spuren voreingestellt. Dann gibt es die Laufwerke C und D, die auf 8 Zoll eingestellt sind. C ist die Vorderseite eines solchen Laufwerkes mit 243 KByte Kapazität. D ist die Rückseite mit weiteren 243 KByte. Das Format ist so gewählt, daß man (z. B. aus Amerika) CP/M-Software direkt bestellen kann. Man kann sie ohne Probleme auf das 80-Spur-Format kopieren. Das 8-Zoll-Format ist in Amerika sehr verbreitet, alle Softwarehäuser bieten Programme für CP/M-80 auch auf 8-Zoll-Disketten an. Die Bezeichnung lautet 8 Zoll, IBM, einfache Dichte. Diese Angabe kennzeichnet den einzigen Standard, der bei Floppys existiert. CP/M ist nämlich in mancher Hinsicht auch schon wieder zu flexibel geworden. So kann man bestimmen, wie viele Namenseinträge das Inhaltsverzeichnis haben soll, wie viele Sektoren eine Spur haben

soll, wo das Inhaltsverzeichnis liegen soll, ob Zwischenräume (SKEW) bei den Sektoren verwendet werden sollen und vieles mehr. Man kann sagen, daß es pro Computer-Hersteller mindestens drei verschiedene Diskettenformate gibt, wenn es nicht IBM-Format ist.

Das hängt damit zusammen, daß viele Entwicklungen unabhängig voneinander verlaufen und viele Lösungen für das eine Problem existieren, wie man Daten auf die Disketten-Scheibe abspeichern kann.

Die Disk-Routinen im FLOMON

Ein wichtiger Hinweis zum Aufruf der Floppy-Unterprogramme im Monitor. Es gibt beim FLOMON drei Einsprünge. MINI und MAXI sind kompatibel zum alten mc-Monitor, der Einsprung FLOPPY ist der modernere, den man in Zukunft für eigene Anwendungen verwenden sollte. Im HL-Register muß vor dem Aufruf die Hauptspeicherquell- oder -zieladresse der Daten, die einen Sektor füllen, stehen. Im Register D muß die Spuradresse stehen, normalerweise numeriert ab 0. Im Register E steht der Sektor, normalerweise numeriert ab 1. Das Register B enthält den Befehl in verschlüsselter Form. Wenn B = 1, dann soll eine Leseoperation durchgeführt werden. Wenn B = 2, so soll auf die Diskette geschrieben werden. Wenn B = 0, ist eine Sonderfunktion gemeint. Im Register D muß die Steprate stehen, also ein Zahl von 0 bis 3, die die Schrittfrequenz des Schrittmotors im Laufwerk beim Spurwechsel steuert. Der Wert 0 ergibt die höchste Frequenz, 3 die langsamste (siehe FLO2-Beschreibung).

Wenn man in Register D Bit 7 setzt, so wird bei einem Zugriff auf die Floppy-Rückseite auch das Rückseitenbit im Format auf 1 geprüft (sonst auf 0). Wenn man den Einsprung MINI verwendet, wird immer auf 1 geprüft.

Im Register C muß der Laufwerkscode stehen. Dafür gilt folgende Belegung:

7	6	5	4	3	2	1	0
sso	mot	min	dens	d	d	d	d

sso bestimmt die Seite des Laufwerks. sso = 0 ist die Vorderseite. Mit mot = 1 kann man den Laufwerksmotor ausschalten. Eine andere Möglichkeit den Motor automatisch auszuschalten, besteht aber z. B. darin, ein Monoflop einzubauen, das immer bei Head-Load getriggert wird und nach einiger Zeit abfällt. Man muß dann aber auch die Leitung READY mit einem zweiten Monoflop bedienen, so daß das Laufwerk erst dann READY meldet, wenn der Motor seine volle Drehzahl wieder erreicht hat. min und dens wählen die Dichte aus und die Bits d das Laufwerk (siehe FLO-2-Beschreibung).

```
E>c:stat e:dsk:

    E: Drive Characteristics
 1440: 128 Byte Record Capacity
  180: Kilobyte Drive Capacity
   64: 32 Byte Directory Entries
   64: Checked Directory Entries
  128: Records/ Extent
    8: Records/ Block
   15: Sectors/ Track
    0: Reserved Tracks
```

Abb. 8.6.13 Eine RAM-Floppy ist vorgesehen. Als Laufwerk E wird sie angesprochen

RAM-Floppy ist vorgesehen

Das Laufwerk E bietet noch eine Besonderheit, die RAM-Floppy. Wenn man mehrere Speicherbänke, z. B. insgesamt 256 KByte, besitzt, so kann man die über 64 KByte liegenden Bänke als RAM-Floppy nutzen. Dann hat man ein zusätzliches Laufwerk, das sehr sehr schnell ist. *Abb. 8.6.13* zeigt einen Ausdruck des Inhaltsverzeichnisses einer solchen RAM-Floppy-Karte mit

```
                ;***********************************************
                ;* COLD - BOOT PROGRAMM FUER MINILAUFWERKE     *
                ;* ROLF-DIETER KLEIN 841204     1.0            *
                ;* 80 SPUR LAUFWERKE   1K PRO SEKTOR BOOT      *
                ;***********************************************

                ; DAS PROGRAMM WIRD AUF SEKTOR 1 TRACK 0 ABGELEGT

FC00                    ORG OFC00H                ;ANFANGSADRESSE, BEI DD

A000 =          OFFSET  EQU      0A000H           ;
D400 =          CPMB    EQU      03400H+OFFSET    ;START CP/M
EA00 =          BIOS    EQU      04A00H+OFFSET    ;START BIOS

                WBOOTO:

FC00 310001             LXI SP,100H      ;START OF STACK
                ;
FC03 2100D4             LXI H,CPMB       ;START ADRESSE
FC06 0607               MVI B,7          ;ANZAHL DER SEKTOREN  A 1024 BYTES
                                         ; FUELLT GENAU BIS OEFFF
                                         ;1C00H BYTES (GROESSER ALS STD 34H/8)
FC08 1600               MVI D,0          ;START BEI TRACK 0 SEKTOR 2, DA 1 BOOTSEKTOR
FC0A 1E02               MVI E,2          ;1..5 SEKTOR
                RDSEC:
FC0C E5                 PUSH H
FC0D D5                 PUSH D
FC0E C5                 PUSH B
FC0F 0ED0               MVI C,11010000B ;LAUFWERK 0 DOUBLE DENSE MINI
FC11 0601               MVI B,1          ;READ  BEI SYSGEN 2 SETZEN
FC13 CD27F0             CALL OF027H
FC16 C1                 POP B
FC17 D1                 POP D
FC18 E1                 POP H
FC19 DA1EF0             JC OF01EH        ;MONITOR AUFRUFEN BEI FEHLER
FC1C D5                 PUSH D           ;NEXT ADRESSE
FC1D 110004             LXI D,1024       ;DOUBLE DENSE BLOECKE
FC20 19                 DAD D
FC21 D1                 POP D
FC22 1C                 INR E            ;SEKTORANZAHL + 1
FC23 7B                 MOV A,E
FC24 FE06               CPI 5+1          ;1..5  ERST BEI 6 NEUE SPUR (4 MAX)
FC26 DA2CFC             JC RD1
FC29 1E01               MVI E,1          ;START OVER
FC2B 14                 INR D
FC2C 05         RD1:    DCR B
FC2D C20CFC             JNZ RDSEC
FC30 C300EA             JMP BIOS         ;START COLDBIOS

FC33                    END
```

Abb. 8.6.14 Das BOOT-Programm lädt das System in den Speicher

Beispielprogrammen. Das Programm prüft, wieviel RAM vorhanden ist. Man kann also auch weniger Speicher bereithalten. Wenn man dann mehr auf der RAM-Floppy abspeichern will, als RAM vorhanden ist, erscheint die Fehlermeldung „BAD SECTOR". Im BIOS könnte man aber auch eine genauere Anpassung der Kapazität durchführen.

Man besitzt 60 KByte auf der RAM-Floppy, wenn man eine zusätzliche Speicherbank auf Adresse 10000h bis 1FFFFh legt, 120 KByte mit der nächsten Bank und 180 KByte mit drei Speicherbänken, von 10000 bis 3FFFFh. Im Bereich 0 bis FFFFh benötigt man in jedem Fall RAM.

Der Urlader

Abb. 8.6.14 zeigt das sogenannte BOOT-Programm. Es befindet sich bei uns auf dem ersten Sektor in Spur 0. Beim Start durch Flomon wird es auf Adresse FC00h geladen und dort von FLOMON gestartet. Das Programm liest dann die restlichen Sektoren, es sind 7 zu je 1024 Bytes, in den Speicher und startet das Programm, also das CP/M. Achtung, der Rest des Boot-Sektors bleibt leer und wird nicht verwendet. Die Rückseite der beidseitig beschreibbaren Diskette wird nicht fürs System verwendet, damit jemand mit Laufwerken, die nur auf einer Seite arbeiten können, unsere Originaldisketten lesen kann.

Auf der CP/M-Diskette des Franzis-Software-Service befindet sich auch ein verändertes SYSGEN-Programm mit dem Namen SYSGEN80. Damit kann man Sicherheitskopien von CP/M herstellen. Das Programm ist speziell angepaßt. *Abb. 8.6.15* zeigt ein Beispiel. *Abb. 8.6.16* zeigt ein kleines Programm, mit dem die Steprate umgestellt werden kann.

Das Formatieren

Abb. 8.6.17 zeigt das Listing eines Formatierers. Mit diesem Universal-Formatierer lassen sich alle gängigen und manche exotischen Formate herstellen, natürlich auch unser mc-Format. Das Programm läuft auch ohne CP/M, denn es verwendet ausschließlich die Einsprünge von FLOMON (oder mc-Monitor).

```
A>sysgen80
SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP)a
SOURCE ON A, THEN TYPE RETURN
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)a
DESTINATION ON A, THEN TYPE RETURN
```

Abb. 8.6.15 SYSGEN80-Bedienung

```
                    ;************************************
                    ;* Stepraten Einstellung, diese     *
                    ;* bleibt bis zum naechsten Kalt-   *
                    ;* Start erhalten                   *
                    ;* Alle Laufwerke sind betroffen    *
                    ;* Rolf-Dieter Klein 841220   1.0   *
                    ;************************************

                    .z80

F027                mini    equ     0f027h

0000'               start:
0000'   06 00               ld b,0  ; Stepraten-Befehl
0002'   16 00               ld d,0  ; 0=maximale Steprate (0..3)
0004'   CD F027             call mini       ; und setzen
0007'   C3 0000             jp 0    ; anschliessend Warm-Start
                            end
```

Abb. 8.6.16 So wird die Steprate eingestellt

```
rom  abs                                                        checksum
0100  C3 00 07 C3 00 07 C3 09 F0 C3 03 F0 F1 DB 40 DB   += 07ED
0110  30 DB C0 FB ED 4D 21 07 15 CD 75 0B DB 40 DB 30   += 07B0
0120  DB C0 ED 56 3E C3 32 38 00 21 0C 01 22 39 00 CD   += 059F
0130  BE 04 AF 32 61 1B 3A 5B 1B 47 C5 CD F3 01 CD E3   += 0716
0140  02 CD 66 06 CD 45 02 3A 5C 1B FE 01 C2 7B 01 3A   += 0571
0150  5E 1B F6 80 32 5E 1B 3A 60 1B B7 CA 63 01 3E 80   += 05E9
0160  32 5F 1B CD E3 02 CD 66 06 CD 45 02 3A 5E 1B E6   += 063E
0170  7F 32 5E 1B AF 32 5F 1B 3A 61 1B 3C 32 61 1B 47   += 0460
0180  3A 5B 1B BB CA BA 01 CD 46 05 0E 46 CD 06 01 C1   += 05BB
0190  10 AB 21 D3 17 CD 75 0B CD BE 04 AF 32 61 1B 3A   += 0603
01A0  5B 1B 47 C5 CD F3 01 CD 62 04 3A 5C 1B FE 01 C2   += 06E2
01B0  D5 01 3A 5E 1B F6 80 32 5E 1B 3A 60 1B B7 CA C6   += 069D
01C0  01 3E 80 32 5F 1B CD 62 04 3A 5E 1B E6 7F 32 5E   += 0540
01D0  1B AF 32 5F 1B 3A 61 1B 3C 32 61 1B 47 3A 5B 1B   += 03FE
01E0  BB CA E7 01 CD 46 05 0E 56 CD 06 01 C1 10 B4 CD   += 070C
01F0  BE 04 C9 3A 55 1B B7 C8 AF 32 54 1B 3A 61 1B B7   += 063B
0200  C2 25 02 21 80 00 22 56 1B 3E 00 32 59 1B 3E 00   += 0339
0210  32 5C 1B 3E 10 32 5D 1B 3E 1B 32 5A 1B 3E 28 32   += 0330
0220  5B 1B C3 44 02 21 00 01 22 56 1B 3E 01 32 59 1B   += 0310
0230  3E 00 32 5C 1B 3E 10 32 5D 1B 3E 36 32 5A 1B 3E   += 032F
0240  28 32 5B 1B C9 B7 CB F5 21 BC 16 CD 75 0B 3A 61   += 06E5
0250  1B 6F 26 00 CD AE 02 21 C7 16 CD 75 0B 3A 62 1B   += 0529
0260  6F 26 00 CD AE 02 CD 1F 0D F1 CB 57 21 BB 16 CA   += 06A7
0270  75 02 21 16 16 CB 5F CA 7D 02 21 DE 15 CB 67 CA   += 0647
0280  B5 02 21 A7 15 CB 6F CA BD 02 21 70 15 CB 77 CA   += 06A9
0290  95 02 21 39 15 CA 9B 02 21 39 15 CD 75 0B CD 09   += 04FF
02A0  01 FE 57 CA AB 02 FE 77 C2 9E 02 C3 00 07 AF F5   += 0812
02B0  3E 10 01 0A 00 11 00 00 EB F5 CB 23 CB 12 CB 15   += 04F5
02C0  CB 14 CB C3 AF ED 42 D2 CD 02 09 CB B3 F1 3D 20   += 0B91
02D0  EB EB 7B F6 30 F5 7C B5 20 D6 F1 B7 CB 4F CD 06   += 0A22
02E0  01 1B F7 DD 21 63 1B 3A 59 1B B7 C2 F5 02 06 10   += 05BA
02F0  0E FF C3 F9 02 06 20 0E 4E DD 71 00 DD 23 05 C2   += 0662
0300  F9 02 3E 01 32 62 1B 3A 5D 1B 47 C5 3A 59 1B B7   += 0503
0310  C2 B2 03 06 06 DD 36 00 00 DD 23 05 C2 15 03 DD   += 0522
0320  36 00 FE DD 23 3A 61 1B DD 77 00 DD 23 3A 5F 1B   += 05EC
0330  E6 80 B7 C2 3D 03 DD 36 00 00 C3 41 03 DD 36 00   += 064C
0340  01 DD 23 3A 62 1B DD 77 00 DD 23 2A 56 1B 7C FE   += 061B
```

zu Abb. 8.6.17

```
0350  04 C2 56 03 3E 03 DD 77 00 DD 23 DD 36 00 F7 DD  += 069B
0360  23 06 0B DD 36 00 FF DD 23 05 C2 63 03 06 06 DD  += 055C
0370  36 00 00 DD 23 05 C2 6F 03 DD 36 00 FB DD 23 C3  += 0640
0380  06 04 06 0C DD 36 00 00 DD 23 05 C2 84 03 06 03  += 0386
0390  DD 36 00 F5 DD 23 05 C2 90 03 DD 36 00 FE DD 23  += 0773
03A0  3A 61 18 DD 77 00 DD 23 3A 5F 18 E6 80 B7 C2 B8  += 074F
03B0  03 DD 36 00 00 C3 BC 03 DD 36 00 01 DD 23 3A 62  += 054B
03C0  18 DD 77 00 DD 23 2A 56 18 7C FE 04 C2 D1 03 3E  += 0656
03D0  03 DD 77 00 DD 23 DD 36 00 F7 DD 23 06 16 DD 36  += 0690
03E0  00 4E DD 23 05 C2 DE 03 06 0C DD 36 00 00 DD 23  += 051B
03F0  05 C2 EA 03 06 03 DD 36 00 F5 DD 23 05 C2 F6 03  += 0685
0400  DD 36 00 FB DD 23 3A 58 18 4F 2A 56 18 DD 71 00  += 05ED
0410  DD 23 2B 7C B5 C2 0D 04 DD 36 00 F7 DD 23 3A 59  += 06CC
0420  18 B7 C2 2A 04 0E FF C3 2C 04 0E 4E 3A 5A 18 47  += 050E
0430  DD 71 00 DD 23 05 C2 30 04 3A 62 18 3C 32 62 18  += 04E5
0440  C1 05 C2 0B 03 3A 59 18 B7 C2 51 04 0E FF C3 53  += 0632
0450  04 0E 4E 21 E8 03 DD 71 00 DD 23 2B 7C B5 C2 56  += 062E
0460  04 C9 CD 82 04 3E 01 32 62 18 3A 5D 18 47 C5 CD  += 0593
0470  76 05 CD 45 02 3A 62 18 3C 32 62 18 C1 05 C2 6E  += 0521
0480  04 C9 C5 01 05 0D 0B 79 B0 C2 86 04 C1 C9 3A 53  += 063C
0490  18 B7 C2 C6 04 21 FA 00 E5 DB C0 CD 82 04 FB CD  += 0911
04A0  2E 05 F3 E1 DB C0 E6 80 CA 2D 05 2B 7C B5 20 E8  += 0868
04B0  21 D6 16 CD 75 0B CD 09 01 FE 57 CA C3 04 FE 77  += 078C
04C0  C2 B6 04 C3 00 07 FE 01 C2 FC 04 21 FA 00 E5 DB  += 07E2
04D0  40 CD 82 04 FB CD 36 05 F3 E1 DB 40 E6 80 CA 2D  += 08E2
04E0  05 2B 7C B5 20 E8 21 D6 16 CD 75 0B CD 09 01 FE  += 069B
04F0  57 CA F9 04 FE 77 C2 EC 04 C3 00 07 21 FA 00 E5  += 080F
0500  DB 30 CD 82 04 FB CD 3E 05 F3 E1 DB 30 E6 80 CA  += 097B
0510  2D 05 2B 7C B5 20 E8 21 D6 16 CD 75 0B CD 09 01  += 05C7
0520  FE 57 CA 2A 05 FE 77 C2 1D 05 C3 00 07 C9 CD 23  += 072A
0530  06 3E 0B D3 C0 76 CD 23 06 3E 0B D3 40 76 CD 23  += 0610
0540  06 3E 0B D3 30 76 CD 82 04 FB CD 4F 05 F3 C9 CD  += 07C0
0550  23 06 3A 53 18 B7 C2 60 05 3E 5B D3 C0 C3 75 05  += 0615
0560  FE 01 C2 6C 05 3E 5B D3 40 C3 75 05 FE 02 C2 75  += 0752
0570  05 3E 5B D3 30 76 06 02 C5 FB CD A2 05 F3 C1 3A  += 0741
0580  53 18 B7 C2 8B 05 DB C0 C3 9C 05 FE 01 C2 95 05  += 07CE
0590  DB 40 C3 9C 05 FE 02 C2 9C 05 DB 30 E6 3C C8 10  += 07E7
05A0  D7 C9 CD 23 06 3A 53 18 B7 C2 D1 05 3A 62 18 D3  += 0711
05B0  C2 21 63 18 0E C3 3A 5F 18 E6 80 07 07 E6 02 C6  += 0602
05C0  88 D3 C0 DB C4 07 D2 C3 05 ED A2 C3 C3 05 C3 22  += 095A
05D0  06 FE 01 C2 FB 05 3A 62 18 D3 42 21 63 18 0E 43  += 057D
05E0  3A 5F 18 E6 80 07 07 E6 02 C6 88 D3 40 DB 44 07  += 0694
05F0  D2 ED 05 ED A2 C3 ED 05 C3 22 06 FE 02 C2 22 06  += 07DD
0600  3A 62 18 D3 32 21 63 18 0E 33 3A 5F 18 E6 80 07  += 04B4
0610  07 E6 02 C6 88 D3 30 DB 34 07 D2 17 06 ED A2 C3  += 0797
0620  17 06 C9 C5 3A 5E 18 47 3A 59 18 B7 C2 33 06 78  += 0577
0630  F6 10 47 3A 54 18 B7 C2 3E 06 78 F6 20 47 3A 53  += 0612
0640  18 B7 C2 4C 06 78 C1 D3 C4 C3 65 06 FE 01 C2 5A  += 07FC
0650  06 78 C1 EE 10 D3 44 C3 65 06 FE 02 C2 65 06 78  += 0727
0660  C1 EE 10 D3 34 C9 CD 82 04 FB CD BE 06 F3 3A 53  += 08BE
0670  18 B7 C2 7A 06 DB C0 C3 8B 06 FE 01 C2 84 06 DB  += 0826
0680  40 C3 8B 06 FE 02 C2 8B 06 DB 30 E6 44 C9 CD 23  += 07D5
0690  06 3A 53 18 B7 C2 B8 06 21 63 18 0E C3 3A 5F 18  += 0500
06A0  E6 80 07 07 E6 02 F6 F4 D3 C0 DB C4 07 D2 AA 06  += 0901
06B0  ED A3 C3 AA 06 C3 FF 06 FE 01 C2 DD 06 21 63 18  += 080B
06C0  0E 43 3A 5F 18 E6 80 07 07 E6 02 F6 F4 D3 40 DB  += 0736
06D0  44 07 D2 CF 06 ED A3 C3 CF 06 C3 FF 06 FE 02 C2  += 08A4
06E0  FF 06 21 63 18 0E 33 3A 5F 18 E6 80 07 07 E6 02  += 04EF
06F0  F6 F4 D3 30 DB 34 07 D2 F4 06 ED A3 C3 F4 06 C9  += 09E5
0700  21 CA 0E CD 75 0B CD 09 01 FE 37 D2 06 07 FE 31  += 0660
0710  DA 06 07 F5 AF 32 55 18 32 53 18 3E E5 32 58 18  += 058C
0720  AF 32 5F 18 32 60 18 F1 FE 31 C2 40 07 AF 32 54  += 0660
0730  18 CD 0F 09 CD 55 09 CD 90 09 CD B3 09 C3 F1 07  += 06D2
```

zu Abb. 8.6.17

```
0740  FE 32 C2 59 07 3E 01 32 54 18 CD 0F 09 CD 55 09  += 053F
0750  CD 90 09 CD 83 09 C3 F1 07 FE 33 C2 85 07 3E 01  += 0768
0760  32 54 18 21 80 00 22 56 18 3E 00 32 59 18 3E 18  += 0309
0770  32 5A 18 3E 00 32 5C 18 3E 1A 32 5D 18 3E 4D 32  += 0344
0780  5B 18 C3 F1 07 FE 34 C2 BB 07 AF 32 54 18 3E 01  += 066D
0790  32 55 18 32 60 18 21 00 01 22 56 18 3E 01 32 59  += 02C5
07A0  18 3E 00 32 5C 18 3E 10 32 5D 18 3E 36 32 5A 18  += 0309
07B0  3E 28 32 5B 18 C3 F1 07 FE 35 C2 E9 07 3E 00 32  += 061B
07C0  54 18 21 00 04 22 56 18 3E 01 32 59 18 3E 36 32  += 02A9
07D0  5A 18 3E 05 32 5D 18 3E 01 32 5C 18 3E 01 32 60  += 0312
07E0  18 3E 50 32 5B 18 C3 F1 07 FE 36 C2 F1 07 C3 FB  += 07B2
07F0  08 21 80 14 CD 75 08 CD 09 01 FE 34 30 05 FE 31  += 05A7
0800  D2 08 08 FE 0D C2 F7 07 FE 32 C2 12 08 3E 01 C3  += 06BB
0810  1D 08 FE 33 C2 1C 08 3E 02 C3 1D 08 AF 32 53 18  += 04B0
0820  21 A0 13 CD 75 08 CD 09 01 FE 3A D2 26 08 FE 30  += 065E
0830  DA 26 08 FE 39 D2 A2 08 FE 31 DA A2 08 FE 31 C2  += 085F
0840  47 08 3E 01 C3 BA 08 FE 32 C2 51 08 3E 02 C3 BA  += 05BB
0850  08 FE 33 C2 5B 08 3E 04 C3 BA 08 FE 34 C2 65 08  += 0656
0860  3E 08 C3 BA 08 FE 35 C2 6F 08 3E 81 C3 BA 08 FE  += 0719
0870  36 C2 79 08 3E 82 C3 BA 08 FE 37 C2 83 08 3E 84  += 06D2
0880  C3 BA 08 FE 38 C2 BA 08 3E 88 32 5E 18 3A 60 18  += 05FF
0890  B7 CA 9F 08 3A 5E 18 E6 80 CA 9F 08 32 5F 18 C3  += 071B
08A0  BB 08 FE 39 C2 AD 08 C3 00 07 C3 BB 08 FE 30 C2  += 07B1
08B0  BB 08 C3 FB 08 C3 BB 08 C3 20 08 CD 84 08 21 9B  += 070C
08C0  0E CD 75 08 CD 09 01 FE 4A CA E0 08 FE 6A CA E0  += 083E
08D0  08 FE 6E CA E0 08 FE 4E CA E0 08 FE 03 C2 BB 08  += 08AA
08E0  FE 6E CA EA 08 FE 4E C2 ED 08 C3 00 07 FE 03 C2  += 08BB
08F0  F5 08 C3 FB 08 CD 16 01 C3 20 08 21 01 18 CD 75  += 060E
0900  08 CD 09 01 FE 0D C2 01 09 ED 46 CD 00 00 C9 21  += 05A3
0910  64 10 CD 75 08 CD 09 01 FE 36 D2 15 09 FE 31 DA  += 06C5
0920  15 09 FE 31 C2 2C 09 3E 23 C3 51 09 FE 32 C2 36  += 05EA
0930  09 3E 28 C3 51 09 FE 33 C2 40 09 3E 46 C3 51 09  += 0569
0940  FE 34 C2 4A 09 3E 4D C3 51 09 FE 35 C2 51 09 3E  += 067C
0950  50 32 5B 18 C9 21 BA 10 CD 75 08 CD 09 01 FE 34  += 05FF
0960  D2 5B 09 FE 31 DA 5B 09 FE 31 C2 75 09 3E 00 32  += 06B2
0970  5C 18 C3 BC 09 FE 32 C2 82 09 3E 01 32 5C 18 C3  += 05F1
0980  BC 09 3E 01 32 5C 18 3E 01 32 60 18 32 5C 18 C9  += 03D2
0990  21 26 10 CD 75 08 CD 09 01 FE 33 D2 96 09 FE 31  += 064C
09A0  DA 96 09 FE 31 C2 AD 09 3E 00 C3 AF 09 3E 01 32  += 064A
09B0  59 18 C9 3A 54 18 B7 C2 BF 0A 3A 59 18 B7 C2 1C  += 0632
09C0  0A 21 1A 11 CD 75 08 CD 09 01 FE 34 D2 C7 09 FE  += 064C
09D0  31 DA C7 09 FE 31 C2 EC 09 21 80 00 22 56 18 3E  += 0630
09E0  10 32 5D 18 3E 18 32 5A 18 C3 19 0A FE 32 C2 04  += 0490
09F0  0A 21 80 00 22 56 18 3E 12 32 5D 18 3E 0A 32 5A  += 0306
0A00  18 C3 19 0A FE 33 C2 19 0A 21 00 01 22 56 18 3E  += 0404
0A10  0A 32 5D 18 3E 10 32 5A 18 C3 BC 0A 21 9E 11 CD  += 0499
0A20  75 08 CD 09 01 FE 35 D2 22 0A FE 31 DA 22 0A FE  += 06BB
0A30  31 C2 47 0A 21 00 01 22 56 18 3E 10 32 5D 18 3E  += 0329
0A40  36 32 5A 18 C3 BC 0A FE 32 C2 5F 0A 21 00 02 22  += 04D3
0A50  56 18 3E 09 32 5D 18 3E 36 32 5A 18 C3 BC 0A FE  += 04CB
0A60  33 C2 77 0A 21 00 02 22 56 18 3E 0A 32 5D 18 3E  += 0356
0A70  28 32 5A 18 C3 BC 0A FE 34 C2 BC 0A 21 00 04 22  += 04F6
0A80  56 18 3E 05 32 5D 18 3E 36 32 5A 18 C3 74 08 3A  += 03EC
0A90  59 18 B7 C2 D4 0A 21 4C 12 CD 75 08 CD 09 01 FE  += 0669
0AA0  33 D2 9C 0A FE 31 DA 9C 0A FE 31 C2 C1 0A 21 80  += 07B7
0AB0  00 22 56 18 3E 1A 32 5D 18 3E 18 32 5A 18 C3 D1  += 0420
0AC0  0A 21 00 01 22 56 18 3E 0F 32 5D 18 3E 18 32 5A  += 0295
0AD0  18 C3 74 08 21 A4 12 CD 75 08 CD 09 01 FE 37 D2  += 065C
0AE0  DA 0A FE 31 DA DA 0A FE 31 C2 FF 0A 21 00 01 22  += 070F
0AF0  56 18 3E 1A 32 5D 18 3E 36 32 5A 18 C3 74 08 FE  += 04C5
0B00  32 C2 17 08 21 00 02 22 56 18 3E 0E 32 5D 18 3E  += 02FA
0B10  36 32 5A 18 C3 74 08 FE 33 C2 2F 08 21 00 02 22  += 04BE
0B20  56 18 3E 0F 32 5D 18 3E 36 32 5A 18 C3 74 08 FE  += 04BA
0B30  34 C2 47 08 21 00 02 22 56 18 3E 10 32 5D 18 3E  += 032E
0B40  28 32 5A 18 C3 74 08 FE 35 C2 5F 08 21 00 04 22  += 04B4
```

zu Abb. 8.6.17

```
0B50  56 1B 3E 0B 32 5D 1B 3E 36 32 5A 1B C3 74 0B FE   += 04B3
0B60  36 C2 74 0B 21 00 04 22 56 1B 3E 09 32 5D 1B 3E   += 035B
0B70  2B 32 5A 1B C9 7E B7 CA B3 0B 4F CD 06 01 23 7E   += 05E6
0BB0  C3 76 0B C9 0E 1A CD 06 01 3A 54 1B B7 C2 96 0B   += 05C9
0B90  21 45 0D C3 99 0B 21 55 0D CD 75 0B 3A 53 1B B7   += 0506
0BA0  C2 A9 0B 21 65 0D C3 B7 0B FE 01 C2 B4 0B 21 6C   += 069B
0BB0  0D C3 B7 0B 21 79 0D CD 75 0B 3A 59 1B B7 C2 C7   += 0671
0BC0  0B 21 B6 0D C3 CA 0B 21 BB 0D 3A 55 1B FE 01 C2   += 057B
0BD0  D5 0B 21 91 0D CD 75 0B CD 1F 0D CD 1F 0D 21 99   += 059B
0BE0  0D CD 75 0B 3A 5B 1B 6F 26 00 CD AE 02 CD 1F 0D   += 0512
0BF0  21 AD 0D CD 75 0B 3A 5D 1B 6F 26 00 CD AE 02 CD   += 05B6
0C00  1F 0D 21 C1 0D CD 75 0B 2A 56 1B CD AE 02 CD 1F   += 0569
0C10  0D 21 D5 0D CD 75 0B 3A 5E 1B E6 0F FE 01 C2 26   += 05E9
0C20  0C 0E 41 C3 46 0C FE 02 C2 30 0C 0E 42 C3 46 0C   += 04D3
0C30  FE 04 C2 3A 0C 0E 43 C3 46 0C FE 0B C2 44 0C 0E   += 0596
0C40  44 C3 46 0C 0E 3F CD 06 01 CD 1F 0D CD 1F 0D 3A   += 04A6
0C50  5C 1B FE 01 C2 6D 0C 3A 60 1B B7 C2 64 0C 21 39   += 05A3
0C60  0E C3 67 0C 21 5B 0E CD 75 0B C3 91 0C 3A 5E 1B   += 052B
0C70  E6 B0 C2 7E 0C 21 DF 0D CD 75 0B C3 91 0C 3A 60   += 0706
0CB0  1B B7 C2 BB 0C 21 FB 0D C3 BE 0C 21 16 0E CD 75   += 0635
0C90  0B CD 1F 0D CD 1F 0D 21 26 0D CD 75 0B 3A 5D 1B   += 044D
0CA0  6F 26 00 3A 5B 1B 5F 16 00 CD 02 0D 3A 5C 1B FE   += 043F
0CB0  01 C2 BA 0C 11 02 00 CD 02 0D ED 5B 56 1B 7A B7   += 055F
0CC0  C2 D0 0C 06 03 CB 2C CB 1D 05 C2 C5 0C C3 F5 0C   += 06E2
0CD0  7A FE 01 C2 E1 0C CB 2C CB 1D CB 2C CB 1D C3 F5   += 0B9E
0CE0  0C 7A FE 02 C2 EE 0C CB 2C CB 1D C3 F5 0C 7A FE   += 0B5D
0CF0  04 C2 F5 0C 00 CD AE 02 21 3D 0D CD 75 0B CD 1F   += 05EB
0D00  0D C9 3E 10 D5 C1 11 00 00 EB F5 CB 43 CA 11 0D   += 06A1
0D10  09 CB 2C CB 1D CB 1A CB 1B F1 3D 20 ED EB C9 21   += 07C3
0D20  D3 16 CD 75 0B C9 53 70 65 69 63 6B 65 72 6B 61   += 06FE
0D30  70 61 7A 69 74 61 65 74 20 3A 20 20 00 20 4B 42   += 04A9
0D40  79 74 65 73 00 4D 69 6E 69 2D 4C 61 75 66 77 65   += 05E3
0D50  72 6B 20 20 00 4D 61 7B 69 2D 4C 61 75 66 77 65   += 053D
0D60  72 6B 20 20 00 46 4C 4F 32 20 20 00 46 4C 4F 31   += 03B2
0D70  20 2B 34 30 6B 29 20 20 00 46 4C 4F 31 20 2B 33   += 030A
0DB0  30 6B 29 20 20 00 46 4D 20 20 00 4D 46 4D 20 20   += 02F4
0D90  00 45 43 4D 41 37 30 20 00 41 6E 7A 61 6B 6C 20   += 041B
0DA0  53 70 75 72 65 6E 20 20 20 20 3D 20 00 41 6E 7A   += 04B3
0DB0  61 6B 6C 20 53 65 6B 74 6F 72 65 6E 20 20 3D 20   += 053D
0DC0  00 42 79 74 65 73 2F 70 72 6F 20 53 65 6B 74 6F   += 05AD
0DD0  72 20 3D 20 00 4C 61 75 66 77 65 72 6B 20 00 56   += 04A6
0DE0  6F 72 64 65 72 73 65 69 74 65 20 77 69 72 64 20   += 062C
0DF0  66 6F 72 6D 61 74 69 65 72 74 00 52 75 65 63 6B   += 0637
0E00  73 65 69 74 65 20 77 69 72 64 20 66 6F 72 6D 61   += 0625
0E10  74 69 65 72 74 00 52 75 65 63 6B 73 65 69 74 65   += 063C
0E20  20 77 69 72 64 20 6D 69 74 20 53 53 4F 20 66 6F   += 054A
0E30  72 6D 61 74 69 65 72 74 00 62 65 69 64 65 20 53   += 05D4
0E40  65 69 74 65 6E 20 77 65 72 64 65 6E 20 66 6F 72   += 0621
0E50  6D 61 74 69 65 72 74 00 62 65 69 64 65 20 53 65   += 05C7
0E60  69 74 65 6E 20 77 65 72 64 65 6E 20 66 6F 72 6D   += 0629
0E70  61 74 69 65 72 74 0D 0A 53 65 69 74 65 20 31 20   += 050B
0EB0  6D 69 74 20 53 53 4F 2C 20 53 65 69 74 65 20 30   += 04F5
0E90  20 6E 6F 72 6D 61 6C 00 0D 0A 41 43 4B 54 55 4E   += 04B3
0EA0  47 20 44 69 73 6B 65 74 74 65 20 77 69 72 64 20   += 059A
0EB0  66 6F 72 6D 61 74 69 65 72 74 0D 0A 53 74 61 72   += 05EE
0EC0  74 20 3D 20 22 4A 22 0D 0A 00 1A 0D 0A 20 55 6E   += 02AA
0ED0  69 76 65 72 73 61 6C 20 46 6F 72 6D 61 74 69 65   += 064D
0EE0  72 65 72 20 20 56 20 31 2E 32 20 2C 20 52 6F 6C   += 0429
0EF0  66 2D 44 69 65 74 65 72 20 4B 6C 65 69 6E 20 0D   += 0530
0F00  0A 20 2B 43 29 20 31 39 3B 34 2C 20 4D 75 65 6E   += 0395
0F10  63 6B 65 6E 20 0D 0A 0D 0A 20 2D 2D 20 20 20 42   += 030B
0F20  69 74 74 65 20 53 79 73 74 65 6D 64 69 73 6B 65   += 066B
0F30  74 74 65 20 6B 65 72 61 75 73 6E 65 6B 6D 65 6E   += 0670
0F40  20 20 20 20 2D 2D 20 0D 0A 20 2D 2D 20 75 6E 64   += 02F2
0F50  20 7A 75 20 66 6F 72 6D 61 74 69 65 72 65 6E 64   += 062F
```

zu Abb. 8.6.17

```
0F60  65 20 44 69 73 6B 65 74 74 65 20 65 69 6E 6C 65   += 05EF
0F70  67 65 6E 20 2D 2D 20 0D 0A 0D 0A 4D 69 6E 69 6C   += 03FB
0F80  61 75 66 77 65 72 6B 65 20 28 35 20 31 2F 34 22   += 04AD
0F90  29 20 3D 20 31 0D 0A 4D 61 78 69 6C 61 75 66 77   += 049C
0FA0  65 72 6B 65 20 28 3B 22 20 20 20 20 29 20 3D 20   += 036F
0FB0  32 0D 0A 53 74 64 20 20 3B 22 20 53 44 20 53 44   += 037C
0FC0  20 37 37 20 53 70 75 72 20 20 3D 20 33 0D 0A 45   += 03B4
0FD0  43 4D 41 20 37 30 20 44 44 20 53 44 20 34 30 20   += 035B
0FE0  53 70 75 72 20 20 3D 20 34 0D 0A 4E 44 52 2D 20   += 03C3
0FF0  44 44 20 44 53 20 3B 30 20 53 70 75 72 20 20 20   += 03F1
1000  20 20 3D 20 35 0D 0A 45 4E 44 45 20 20 20 20 20   += 02A5
1010  20 20 20 20 20 20 20 20 20 20 20 20 20 20 3D 20   += 021D
1020  36 0D 0A 0D 0A 00 1A 0D 0A 65 69 6E 66 61 63 6B   += 0363
1030  65 20 53 63 6B 72 65 69 62 64 69 63 6B 74 65 20   += 05D6
1040  3D 20 31 0D 0A 64 6F 70 70 65 6C 74 65 20 53 63   += 04DB
1050  6B 72 65 69 62 64 69 63 6B 74 65 20 3D 20 32 0D   += 0537
1060  0A 0D 0A 00 1A 0D 0A 33 35 20 53 70 75 72 65 6E   += 0357
1070  20 20 3D 20 31 0D 0A 34 30 20 53 70 75 72 65 6E   += 03E6
1080  20 20 3D 20 32 0D 0A 37 30 20 53 70 75 72 65 6E   += 03EA
1090  20 20 3D 20 33 0D 0A 37 37 20 53 70 75 72 65 6E   += 03F2
10A0  20 20 3D 20 34 0D 0A 38 30 20 53 70 75 72 65 6E   += 03ED
10B0  20 20 3D 20 35 0D 0A 0D 0A 00 1A 0D 0A 45 69 6E   += 024D
10C0  73 65 69 74 69 67 65 73 20 4C 61 75 66 77 65 72   += 0653
10D0  6B 20 20 20 20 20 3D 20 31 0D 0A 44 6F 70 70 65   += 03AB
10E0  6C 74 73 65 69 74 69 67 65 73 20 4C 61 75 66 77   += 065C
10F0  65 72 6B 20 3D 20 32 0D 0A 44 6F 70 70 65 6C 74   += 04E0
1100  73 65 69 74 69 67 20 75 6E 64 20 53 53 4F 20 20   += 0541
1110  20 20 3D 20 33 0D 0A 0D 0A 00 1A 0D 0A 31 32 38   += 01CA
1120  20 42 79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F   += 05BE
1130  72 20 20 28 31 36 20 70 72 6F 20 54 72 61 63 6B   += 04C7
1140  29 20 3D 20 31 0D 0A 31 32 38 20 42 79 74 65 73   += 03B0
1150  20 70 72 6F 20 53 65 6B 74 6F 72 20 20 28 31 38   += 04DA
1160  20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 32 0D   += 046B
1170  0A 32 35 36 20 42 79 74 65 73 20 70 72 6F 20 53   += 04B2
1180  65 6B 74 6F 72 20 20 28 31 30 20 70 72 6F 20 54   += 04D3
1190  72 61 63 6B 29 20 3D 20 33 0D 0A 0D 0A 00 1A 0D   += 02CF
11A0  0A 32 35 36 20 42 79 74 65 73 20 70 72 6F 20 53   += 04B2
11B0  65 6B 74 6F 72 20 20 28 31 36 20 70 72 6F 20 54   += 04D9
11C0  72 61 63 6B 29 20 3D 20 31 0D 0A 35 31 32 20 42   += 03B9
11D0  79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F 72 20   += 05EE
11E0  20 28 39 20 70 72 6F 20 54 72 61 63 6B 29 20 20   += 0470
11F0  3D 20 32 0D 0A 35 31 32 20 42 79 74 65 73 20 70   += 03F5
1200  72 6F 20 53 65 6B 74 6F 72 20 20 28 31 30 20 70   += 04D2
1210  72 6F 20 54 72 61 63 6B 29 20 3D 20 33 0D 0A 31   += 0417
1220  30 32 34 20 42 79 74 65 73 20 70 72 6F 20 53 65   += 0506
1230  6B 74 6F 72 20 28 35 20 70 72 6F 20 54 72 61 63   += 055B
1240  6B 29 20 20 3D 20 34 0D 0A 0D 0A 00 1A 0D 0A 31   += 01F5
1250  32 38 20 42 79 74 65 73 20 70 72 6F 20 53 65 6B   += 0545
1260  74 6F 72 20 28 32 36 20 70 72 6F 20 54 72 61 63   += 0520
1270  6B 29 20 3D 20 31 0D 0A 32 35 36 20 42 79 74 65   += 03AA
1280  73 20 70 72 6F 20 53 65 6B 74 6F 72 20 28 31 35   += 052A
1290  20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 32 0D   += 046B
12A0  0A 0D 0A 00 1A 0D 0A 32 35 36 20 42 79 74 65 73   += 0316
12B0  20 70 72 6F 20 53 65 6B 74 6F 72 20 28 32 36 20   += 04D9
12C0  70 72 6F 20 54 72 61 63 6B 29 20 3D 20 31 0D 0A   += 0454
12D0  35 31 32 20 42 79 74 65 73 20 70 72 6F 20 53 65   += 050B
12E0  6B 74 6F 72 20 28 31 34 20 70 72 6F 20 54 72 61   += 0525
12F0  63 6B 29 20 3D 20 32 0D 0A 35 31 32 20 42 79 74   += 03A4
1300  65 73 20 70 72 6F 20 53 65 6B 74 6F 72 20 28 31   += 055A
1310  35 20 70 72 6F 20 54 72 61 63 6B 29 20 3D 20 33   += 0494
1320  0D 0A 35 31 32 20 42 79 74 65 73 20 70 72 6F 20   += 0467
1330  53 65 6B 74 6F 72 20 28 31 36 20 70 72 6F 20 54   += 050C
1340  72 61 63 6B 29 20 3D 20 34 0D 0A 31 30 32 34 20   += 0379
1350  42 79 74 65 73 20 70 72 6F 20 53 65 6B 74 6F 72   += 0610
1360  20 28 38 20 70 72 6F 20 54 72 61 63 6B 29 20 3D   += 04BC
1370  20 35 0D 0A 31 30 32 34 20 42 79 74 65 73 20 70   += 03EA
```

zu Abb. 8.6.17

```
1380  72 6F 20 53 65 6B 74 6F 72 20 28 39 20 70 72 6F   += 056B
1390  20 54 72 61 63 6B 29 20 3D 20 36 0D 0A 0D 0A 00   += 031F
13A0  1A 0D 0A 4C 61 75 66 77 65 72 6B 20 20 20 20 20   += 0412
13B0  20 20 20 20 20 20 20 20 20 41 20 3D 20 31 0D 0A   += 0226
13C0  4C 61 75 66 77 65 72 6B 20 20 20 20 20 20 20 20   += 0441
13D0  20 20 20 20 20 20 42 20 3D 20 32 0D 0A 4C 61 75   += 02EA
13E0  66 77 65 72 6B 20 20 20 20 20 20 20 20 20 20 20   += 037F
13F0  20 20 20 43 20 3D 20 33 0D 0A 4C 61 75 66 77 65   += 03CE
1400  72 6B 20 20 20 20 20 20 20 20 20 20 20 20 20 20   += 029D
1410  44 20 3D 20 34 0D 0A 4C 61 75 66 77 65 72 6B 20   += 046D
1420  41 20 52 75 65 63 6B 73 65 69 74 65 20 41 20 3D   += 0533
1430  20 35 0D 0A 4C 61 75 66 77 65 72 6B 20 42 20 52   += 04B1
1440  75 65 63 6B 73 65 69 74 65 20 42 20 3D 20 36 0D   += 04E4
1450  0A 4C 61 75 66 77 65 72 6B 20 43 20 52 75 65 63   += 055D
1460  6B 73 65 69 74 65 20 43 20 3D 20 37 0D 0A 4C 61   += 0460
1470  75 66 77 65 72 6B 20 44 20 52 75 65 63 6B 73 65   += 05EA
1480  69 74 65 20 44 20 3D 20 38 0D 0A 4E 45 55 53 54   += 0401
1490  41 52 54 20 20 20 20 3D 20 39 0D 0A 45 4E 44 45   += 0330
14A0  20 20 20 20 20 20 20 20 3D 20 30 0D 0A 0D 0A 00   += 01BB
14B0  1A 0D 0A 46 4C 4F 32 2D 4D 61 78 69 2F 4D 69 6E   += 0453
14C0  69 20 28 30 43 30 6B 29 20 3D 20 31 0D 0A 46 4C   += 033C
14D0  4F 31 2D 4D 61 78 69 20 28 34 30 6B 29 20 20 20   += 03D9
14E0  20 20 20 20 3D 20 32 0D 0A 46 4C 4F 31 2D 4D 69   += 031B
14F0  6E 69 20 28 33 30 6B 29 20 20 20 20 20 20 20 3D   += 0330
1500  20 33 0D 0A 0D 0A 00 1A 0D 0A 44 69 73 6B 65 74   += 0316
1510  74 65 20 77 69 72 64 20 66 6F 72 6D 61 74 69 65   += 0626
1520  72 74 20 2E 2E 2E 20 62 69 74 74 65 20 77 61 72   += 0532
1530  74 65 6E 20 0D 0A 0D 0A 00 0D 0A 53 63 68 72 65   += 03A1
1540  69 62 73 63 68 75 74 7A 20 67 65 73 65 74 7A 74   += 0692
1550  0D 0A 4E 65 75 73 74 61 72 74 20 3D 20 54 61 73   += 0512
1560  74 65 20 57 20 64 72 75 65 63 6B 65 6E 0D 0A 00   += 04DB
1570  0D 0A 52 65 63 6F 72 64 20 54 79 70 20 69 73 74   += 0543
1580  20 66 61 6C 73 63 68 0D 0A 4E 65 75 73 74 61 72   += 058A
1590  74 20 3D 20 54 61 73 74 65 20 57 20 64 72 75 65   += 0539
15A0  63 6B 65 6E 0D 0A 00 0D 0A 52 65 63 6F 72 64 20   += 044E
15B0  6E 69 63 68 74 20 67 65 66 75 6E 64 65 6E 0D 0A   += 0599
15C0  4E 65 75 73 74 61 72 74 20 3D 20 54 61 73 74 65   += 05D4
15D0  20 57 20 64 72 75 65 63 6B 65 6E 0D 0A 00 0D 0A   += 0416
15E0  43 52 43 20 46 65 68 6C 65 72 20 61 75 66 67 65   += 0576
15F0  74 72 65 74 65 6E 0D 0A 4E 65 75 73 74 61 72 74   += 05FF
1600  20 3D 20 54 61 73 74 65 20 57 20 64 72 75 65 63   += 052B
1610  6B 65 6E 0D 0A 00 0D 0A 43 50 55 20 7A 75 20 6C   += 03EF
1620  61 6E 67 73 61 6D 20 66 75 65 72 20 67 65 77 61   += 060D
1630  65 68 6C 74 65 73 20 46 6F 72 6D 61 74 2D 2D 20   += 05BB
1640  44 61 74 65 6E 76 65 72 6C 75 73 74 0D 0A 6B 6F   += 05EF
1650  65 6B 65 72 65 72 20 54 61 6B 74 20 65 72 66 6F   += 05FB
1660  72 64 65 72 6C 69 63 68 0D 0A 4E 65 75 73 74 61   += 05D4
1670  72 74 20 3D 20 54 61 73 74 65 20 57 20 64 72 75   += 0546
1680  65 63 6B 65 6E 0D 0A 00 0D 0A 46 65 68 6C 65 72   += 04BA
1690  20 61 75 66 67 65 74 72 65 74 65 6E 2E 20 4E 65   += 05BB
16A0  75 73 74 61 72 74 20 3D 20 54 61 73 74 65 20 57   += 059B
16B0  20 64 72 75 65 63 6B 65 6E 0D 0A 00 0D 0A 54 72   += 0465
16C0  61 63 6B 20 3D 20 00 2C 20 53 65 6B 74 6F 72 20   += 0490
16D0  3D 20 00 0D 0A 00 0D 0A 4C 61 75 66 77 65 72 6B   += 03CC
16E0  20 6E 69 63 68 74 20 52 45 41 44 59 20 2E 2E 2E   += 0475
16F0  20 62 69 74 74 65 20 70 72 75 65 66 65 6E 3A 0D   += 0594
1700  0A 20 31 2E 20 44 69 73 6B 65 74 74 65 20 66 61   += 04CD
1710  6C 73 63 68 6B 65 72 75 6D 20 65 69 6E 67 65 6C   += 065F
1720  65 67 74 0D 0A 20 32 2E 20 4C 61 75 66 77 65 72   += 04CD
1730  6B 73 74 75 65 72 65 20 6E 69 63 68 74 20 67 65   += 0625
1740  73 63 68 6C 6F 73 73 65 6E 0D 0A 20 33 2E 20 66   += 04F0
1750  61 6C 73 63 68 65 20 4C 61 75 66 77 65 72 6B 73   += 0644
1760  6E 75 6D 6D 65 72 20 61 6E 67 65 67 65 62 65 6E   += 0650
1770  0D 0A 20 34 2E 20 4C 61 75 66 77 65 72 6B 73 6D   += 04DA
1780  6F 74 6F 72 20 64 72 65 68 74 20 73 69 63 68 20   += 05E2
```

zu Abb. 8.6.17

```
1790   6E 69 63 68 74 0D 0A 20 35 2E 20 52 45 41 44 59   += 0445
17A0   2D 4C 65 69 74 75 6E 67 20 6E 69 63 68 74 20 6F   += 05CA
17B0   6B 0D 0A 0D 0A 4E 65 75 73 74 61 72 74 20 3D 20   += 046C
17C0   54 61 73 74 65 20 57 20 64 72 75 65 63 6B 65 6E   += 05E9
17D0   0D 0A 00 0D 0A 44 69 73 6B 65 74 74 65 20 77 69   += 046B
17E0   72 64 20 67 65 70 72 75 65 66 74 20 2E 2E 2E 20   += 0522
17F0   62 69 74 74 65 20 77 61 72 74 65 6E 0D 0A 0D 0A   += 04F7
1800   00 1A 0D 0A 53 79 73 74 65 6D 64 69 73 6B 65 74   += 053A
1810   74 65 20 69 6E 20 4C 61 75 66 77 65 72 6B 20 41   += 0592
1820   20 65 69 6E 6C 65 67 65 6E 2C 0D 0A 64 61 6E 6E   += 054B
1830   20 43 52 20 28 63 61 72 72 69 61 67 65 20 72 65   += 0532
1840   74 75 72 6E 29 20 64 72 75 65 63 6B 65 6E 2E 0D   += 059E
1850   0A 00 00 00 00 00 00 00 E5 00 00 00 00 00 00 00   += 00EF
1860   00 00 00 67 65 70 72 75 65 66 74 20 2E 2E 2E 20   += 042C
1870   62 69 74 74 65 20 77 61 72 74 65 6E 0D 0A 0D 0A   += 04F7
1880
```

Abb. 8.6.17 Ein Universal-Formatierer. Er wendet sich direkt an Flomon

Das Listing ist mit einer Prüfsumme versehen, die man zur Kontrolle beim Eintippen verwenden kann. Dabei wird jeweils die Quersumme einer Zeile gebildet. *Abb. 8.6.18* zeigt, wie sich das Programm meldet. Übrigens kann man das Programm auch mit dem Grundprogramm eingeben, z. B. ab Adresse 8800, wenn man noch nicht stolzer Besitzer von CP/M ist (dann muß es vor dem Start aber nach 100h verschoben werden, z. B. mit dem Z80-LDIR-Befehl, und davor muß man die Bank umschalten).

Beispiel:

```
Adresse>   8800             ; freien Platz nehmen
           LD A,80h         ; Bank 0 ohne EPROM anwählen
           OUT (0C8h),A     ; und schalten
           LD HL,8800h      ; Startadresse
           LD DE,100h       ; Zieladresse
           LD BC,länge      ; Programmlänge eintragen
           LDIR             ; transportieren
           JP 100h          ; und starten
```

```
Universal Formatierer  V 1.2 , Rolf-Dieter Klein
(C) 1984, Muenchen

--  Bitte Systemdiskette herausnehmen  --
-- und zu formatierende Diskette einlegen --

Minilaufwerke (5 1/4") = 1
Maxilaufwerke (8"    ) = 2
Std  8" SD SD 77 Spur  = 3
ECMA 70 DD SD 40 Spur  = 4
NDR- DD DS 80 Spur     = 5
ENDE                   = 6
```

Abb. 8.6.18 Das Startmenü des Formatierers

```
FLO2-Maxi/Mini (0C0h) = 1
FLO1-Maxi (40h)       = 2
FLO1-Mini (30h)       = 3
```

Abb. 8.6.19
FLO2 ist ein eigener Menüpunkt

```
Laufwerk                A = 1
Laufwerk                B = 2
Laufwerk                C = 3
Laufwerk                D = 4
Laufwerk A Rueckseite A = 5
Laufwerk B Rueckseite B = 6
Laufwerk C Rueckseite C = 7
Laufwerk D Rueckseite D = 8
NEUSTART     = 9
ENDE         = 0
```

Abb. 8.6.20 Die Laufwerksauswahl

```
Mini-Laufwerk FLO2 MFM

Anzahl Spuren    = 80
Anzahl Sektoren  = 5
Bytes/pro Sektor = 1024
Laufwerk A

beide Seiten werden formatiert
Seite 1 mit SSO, Seite 0 normal

Speicherkapazitaet :  800 KBytes

ACHTUNG Diskette wird formatiert
Start = "J"
```

Abb. 8.6.21 Die Kontrollausgabe

```
Diskette wird formatiert ... bitte warten
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Diskette wird geprueft ... bitte warten

VVVVJ
```

Abb. 8.6.22 Der Formatiervorgang wird so angezeigt

Vor dem Start alles sichern, z. B. auf Kassette, solange CP/M nicht läuft. Wenn man das NDR-Format herstellen will, so wählt man im Menü den Punkt 5 aus. Danach erscheint *Abb. 8.6.19*. Das Programm unterstützt auch die Floppy-Controller des mc-Computers, daher müssen wir hier FLO2, also 1 auswählen. Dann erscheint *Abb. 8.6.20*. Dort wählt man das Laufwerk aus, hier also A oder B; nimmt man A, muß man 1 eingeben. Die Rückseitenauswahl erfolgt automatisch, und ist im Menü nur für Spezialanwendungen verfügbar. Es erscheint die Kontrollausgabe (*Abb. 8.6.21*).

Alle wichtigen Laufwerksdaten werden noch einmal ausgegeben. Wenn man jetzt die Taste „J" zur Bestätigung drückt, beginnt die Formatierung. Achtung, keine Diskette mit Daten einlegen, die werden beim Formatieren unter Garantie gelöscht. Das Formatieren dauert etwa 2 Min. *Abb. 8.6.22* zeigt den Bildschirm während der Formatierung. Die erste Reihe wird beim eigentlichen Formatieren ausgegeben, jedes F steht für eine Spur, bei uns für Vorder- und Rückseite, das V darunter für einen Prüflesevorgang. Tritt ein Fehler auf, wird eine Meldung auf dem Bildschirm mit Fehlerhinweis ausgegeben.

Wichtige Hinweise

Wenn der Hinweis „CPU zu langsam" auftaucht, hat man entweder einen falschen Quarz in der CPU (es müssen 4 MHz CPU-Takt vorliegen, also 8 MHz Quarz) oder man bearbeitet ein Hoch-Dichte-Format, wie z. B. 8 Zoll mit doppelter Dichte. Dort benötigt man eine 6-MHz-CPU.

Wenn man nicht die fertigen Formate im Menü anwählt, muß man weitere Angaben machen. *Abb. 8.6.23* zeigt die Auswahl der Spurenzahl, *Abb. 8.6.24* die Laufwerksart. SS0 bedeutet hier, daß auf der Rückseite das SS0-Bit im Format gesetzt wird (wie es z. B. beim 5¼-Zoll-mc-Format der Fall ist, während 8 Zoll doppelseitig beim mc-Format ohne SS0 formatiert werden muß). *Abb. 8.6.25* zeigt, wie man die Dichte auswählt. Da gibt es wieder eine Ausnahme, das ECMA-70-Format im Hauptmenü formatiert die erste Spur in einfacher Dichte, den Rest in doppelter Dichte. *Abb. 8.6.26* zeigt die Sektorauswahl bei einfacher Dichte 5¼ Zoll, *Abb. 8.6.27* doppelte Dichte bei 5¼ Zoll, *Abb. 8.6.28* einfache Dichte bei 8 Zoll und *Abb. 8.6.29* doppelte Dichte bei 8 Zoll, wofür man aber eine mit 6 MHz betriebene CPU benötigt. Übrigens kann man auch Laufwerke mit 3½ Zoll und 3 Zoll betreiben, da die meisten steckerkompatibel sind. Bei diesen Laufwerken sind zwei Seiten mit je 80 Spuren schon neuer Standard geworden.

```
35 Spuren  = 1
40 Spuren  = 2
70 Spuren  = 3
77 Spuren  = 4
80 Spuren  = 5
```

Abb. 8.6.23 Menü zur Festlegung der Spurenzahl

```
Einseitiges Laufwerk      = 1
Doppeltseitiges Laufwerk = 2
Doppeltseitig und SS0     = 3
```

Abb. 8.6.24 Der Laufwerkstyp muß angegeben werden

```
einfache Schreibdichte = 1
doppelte Schreibdichte = 2
```

Abb. 8.6.25 Die Schreibdichte wird hier festgelegt

```
128 Bytes pro Sektor  (16 pro Track) = 1
128 Bytes pro Sektor  (18 pro Track) = 2
256 Bytes pro Sektor  (10 pro Track) = 3
```

Abb. 8.6.26 Bei einfacher Dichte, 5¼", gibt es diese Auswahl

```
256 Bytes pro Sektor  (16 pro Track) = 1
512 Bytes pro Sektor  (9 pro Track)  = 2
512 Bytes pro Sektor  (10 pro Track) = 3
1024 Bytes pro Sektor (5 pro Track)  = 4
```

Abb. 8.6.27 Bei doppelter Dichte, 5¼", gibt es 4 Wahlmöglichkeiten

```
128 Bytes pro Sektor (26 pro Track) = 1
256 Bytes pro Sektor (15 pro Track) = 2
```

Abb. 8.6.28 Zur einfachen Dichte, 8"

```
256 Bytes pro Sektor (26 pro Track) = 1
512 Bytes pro Sektor (14 pro Track) = 2
512 Bytes pro Sektor (15 pro Track) = 3
512 Bytes pro Sektor (16 pro Track) = 4
1024 Bytes pro Sektor (8 pro Track) = 5
1024 Bytes pro Sektor (9 pro Track) = 6
```

Abb. 8.6.29 Zur doppelten Dichte, 8"

8.6.3 Ausblick

Neben dem schon beschriebenen Betriebssystem CP/M 2.2 kann auch das neuere Betriebssystem CP/M-Plus (3.0) verwendet werden (Einzelheiten dazu vom Elektronikladen, Detmold erhältlich).

Für CP/M gibt es eine Vielzahl von Programmiersprachen: z. B. Forth, C, Algol 60, Fortran, Pascal (Turbo-Pascal), Lisp, Modula 2 und natürlich auch Basic.

Eine besonders gut angepaßte Version, die ständig aktualisiert und erweitert wird, ist das HEBAS* (Version 3.1 auch für CP/M 3.0 und für die CPU 64180). Hierzu ist auch ein kommentiertes Quellisting erhältlich.

* siehe Bezugsquellenverzeichnis

9 Anhang Listings

9.1 Das Scop-Programm

Abb. 9.1.1 zeigt das komplette Listing des Scop-Programms, wie es für den Abgleich der CAS-Baugruppe gebraucht wird.

zu Abb. 9.1.1

```
                              title Oszilloscop-Programm 1.0  Z80  840229

                                    ;************************************
                                    ;* Oscilloscop-Programm   Rev 1.0   *
                                    ;* (C) 1984 Rolf-Dieter Klein       *
                                    ;* Damit ist es moeglich            *
                                    ;* Abgleiche, etc. mit Z80+GDP      *
                                    ;* durchzufuehren. Es wird das      *
                                    ;* Z80-Grundprogramm benoetigt      *
                                    ;* Dieses Programm auf Adr 8800     *
                                    ;* gelegt.                          *
                                    ;************************************

                                    ; IOE-Karte     Adresse 30h
                                    ;
                                    ; Bit 0  Kanal 1  und Triggereingang
                                    ; Bit 1  Kanal 2  fuer Vergleichsmessung

0000'                         aseg

0030                          ioe equ 30h      ; IO-Karte Basis

0000                          test equ 0       ; BASIS des Grundprogs, std=0
                                    ; bei test anders legen (100h)

                              org 8400h        ; Ram-Speicher von 8400 bis 87ff
8400                    synstate:       ds 1
8401                    merker: ds 1    ; merker fuer 1/4 Zaehler
8402                    bcd0:   ds 2    ; messwerte Periodendauer
8404                    bcd1:   ds 2
8406                    buffer: ds 80   ; Speicher fuer Eingaben
```

```
8456                      data:   ds 256+256      ; Datenspeicher fuer Kanale



87FF                              stack equ 87ffh        ; Stack laeuft dort los


                                          ; Konstantendeklarationen fuer Grundprog.

0003                              schreite equ test+3
0006                              drehe  equ test+6
0009                              hebe   equ test+9
000C                              senke  equ test+12
000F                              schleife equ test+15
0012                              endschleife equ test+18
0015                              setze  equ test+21
0018                              moveto  equ test+24
001B                              drawto  equ test+27
001E                              textaus  equ test+30
0021                              textein  equ test+33
0024                              ci  equ test+36
0027                              csts  equ test+39
002A                              ri  equ test+42
002D                              poo  equ test+45
0030                              clrall  equ test+48
0033                              clrinvis equ test+51


0060                              page  equ 60h
0070                              gdp  equ 70h



                                          ; Hauptprogrammbereich


                                  org 8800h      ; Rom-Bereich



8800    C3 8ECD                   jp start

                                          ; Unterprogramme

8803                      wait:           ; warten bis gdp fertig
8803    F5                        push af
8804                      waitl1:
8804    DB 70                     in a,(gdp)
8806    E6 04                     and 4
8808    28 FA                     jr z,waitl1
```

zu Abb. 9.1.1

```
880A    F1                  pop af
880B    C9                  ret


880C               cmd:                ; Befehl an Gdp ausgeben
880C    CD 8803             call wait
880F    D3 70               out (gdp),a
8811    C9                  ret

8812               setpage:                    ; seite in Akku
8812    CD 8803             call wait
8815    E6 F0               and 0f0h           ; wwvv0000
8817    D3 60               out (page),a       ; w= write  v= read
8819    C9                  ret


881A               setpen:
881A    CD 8803             call wait
881D    3E 03               ld a,00000011b
881F    D3 71               out (gdp+1),a
8821    C9                  ret

8822               erapen:
8822    CD 8803             call wait
8825    3E 01               ld a,00000001b
8827    D3 71               out (gdp+1),a
8829    C9                  ret
                                       ;

882A               sync:               ; warten auf vb
882A    DB 70               in a,(gdp)         ; CARRY dann kein sync da
882C    E6 02               and 2
882E    28 0C               jr z,syl
8830    3A 8400             ld a,(synstate)
8833    B7                  or a
8834    37                  scf
8835    C0                  ret nz             ; <>0 dann carry
8836    3C                  inc a
8837    32 8400             ld (synstate),a
883A    AF                  xor a
883B    C9                  ret
883C               syl:
883C    AF                  xor a
883D    32 8400             ld (synstate),a
8840    37                  scf
8841    C9                  ret


                                       ;
8842               gitter:             ; Seite = 3  Gitteraufbau
                                       ; 256 = 5V
                                       ; 10 teile
```

zu Abb. 9.1.1

```
8842    CD 881A                 call setpen
8845    3E F0                   ld a,11110000b  ; schreib,lese =3
8847    CD 8812                 call setpage    ; wait incl.
884A    3E 01                   ld a,00000001b  ; dottet lines
884C    D3 72                   out (gdp+2),a
884E    11 0000                 ld de,0         ; Start
                        ;1      do hl,11        ; incl Start
8851    21 000B                 LD hl,11
8854                    .L1:
8854    E5                      push hl
8855    21 0000                 ld hl,0         ;x=0
8858    CD 0018                 call moveto
885B    21 01FE                 ld hl,510
885E    D5                      push de
885F    CD 001B                 call drawto
8862    D1                      pop de
8863    21 0019                 ld hl,25
8866    19                      add hl,de
8867    EB                      ex de,hl
8868    E1                      pop hl
                        ;1      enddo
8869    2B                      DEC HL
886A    7C                      LD A,H
886B    B5                      OR L
886C    C2 8854                 JP NZ,.L1
                                        ;
886F    21 0000                 ld hl,0
                        ;1      do de,11        ; time
8872    11 000B                 LD de,11
8875                    .L2:
8875    D5                      push de
8876    11 0000                 ld de,0
8879    CD 0018                 call moveto
887C    11 00FA                 ld de,250
887F    E5                      push hl
8880    CD 001B                 call drawto
8883    E1                      pop hl
8884    11 0033                 ld de,51
8887    19                      add hl,de
8888    D1                      pop de
                        ;1      enddo
8889    1B                      DEC DE
888A    7A                      LD A,D
888B    B3                      OR E
888C    C2 8875                 JP NZ,.L2
888F    CD 8803                 call wait
8892    3E 00                   ld a,0
8894    D3 72                   out (gdp+2),a
                                        ;  Mittel-Linien
8896    21 0000                 ld hl,0
8899    11 007D                 ld de,5*25
```

zu Abb. 9.1.1

```
889C    CD 0018                 call moveto
889F    21 01FE                 ld hl,510
88A2    11 007D                 ld de,5*25
88A5    CD 001B                 call drawto
88A8    21 00FF                 ld hl,5*51
88AB    11 0000                 ld de,0
88AE    CD 0018                 call moveto
88B1    21 00FF                 ld hl,5*51
88B4    11 00FA                 ld de,250
88B7    CD 001B                 call drawto
88BA    C9                      ret

                                        ;
88BB                    getframe:               ; einen Datenblock laden
                                        ; Bit 0 = Messport A + Trigger
                                        ; Bit 1 = Messport B

88BB                    get1:
88BB    DB 30                   in a,(ioe)      ; einlesen fuer trigger
88BD    0F                      rrca
88BE    38 FB                   jr c,get1
88C0                    get2:
88C0    DB 30                   in a,(ioe)
88C2    0F                      rrca    ;4T
88C3    30 FB                   jr nc,get2      ;   -----
                                        ;   -----___---- Trigger start
                                        ; 7T
88C5    21 8456                 ld hl,data      ;10T
88C8    06 00                   ld b,0          ; 256 Bytes 7T
88CA    0E 30                   ld c,ioe        ; IOE-Karte 7T
88CC    ED B2                   inir    ; Daten einlesen 11T-Zyklen=5.25ys pro Abtastpunkt
88CE    ED B2                   inir    ; Gesamt 3*256 Bytes
88D0    C9                      ret
                                        ;
                                        ; vom Abtastpunkt bis INIR ca 35T Zyklen = 8.75ys
                                        ;


                                        ;
88D1                    put1frame:              ; ablegen im Bildschirm
88D1    CD 881A                 call setpen
88D4    DD 21 8456              ld ix,data      ; datenquelle
88D8    21 0001                 ld hl,1         ;x=0 ist start
                        ;1      do bc,509
88DB    01 01FD                 LD bc,509
88DE                    .L3:
88DE    11 0082                 ld de,130
88E1    DD 7E 00                ld a,(ix+0)     ; bit 0  hier interessant
88E4    E6 01                   and 1
                        ;2      if nz
88E6    CA 88EC                 JP Z,.L4
```

zu Abb. 9.1.1

```
88E9    11 00B4                  ld de,180
                        ;2       endif
88EC                    .L4:
88EC    CD 0018                  call moveto
88EF    3E 80                    ld a,80h        ; set dot
88F1    CD 880C                  call cmd
88F4    23                       inc hl
88F5    DD 23                    inc ix
                        ;1       enddo
88F7    0B                       DEC BC
88F8    78                       LD A,B
88F9    B1                       OR C
88FA    C2 88DE                  JP NZ,.L3
88FD    C9                       ret

88FE                    put2frame:               ; ablegen im Bildschirm
88FE    CD 881A                  call setpen
8901    DD 21 8456               ld ix,data      ; datenquelle
8905    21 0001                  ld hl,1         ;x=0 ist start
                        ;1       do bc,509
8908    01 01FD                  LD bc,509
890B                    .L5:
890B    11 003C                  ld de,60
890E    DD 7E 00                 ld a,(ix+0)     ; bit 0  hier interessant
8911    E6 02                    and 2   ; kanal 2
                        ;2       if nz
8913    CA 8919                  JP Z,.L6
8916    11 006E                  ld de,110
                        ;2       endif
8919                    .L6:
8919    CD 0018                  call moveto
891C    3E 80                    ld a,80h        ; set dot
891E    CD 880C                  call cmd
8921    23                       inc hl
8922    DD 23                    inc ix
                        ;1       enddo
8924    0B                       DEC BC
8925    78                       LD A,B
8926    B1                       OR C
8927    C2 890B                  JP NZ,.L5
892A    C9                       ret



892B                    clr1frame:               ; da nur zwei linien ist
                                         ; loeschen einfach
892B    CD 8822                  call erapen
892E    21 0001                  ld hl,1
8931    11 0082                  ld de,130
8934    CD 0018                  call moveto
8937    21 01FE                  ld hl,510
```

zu Abb. 9.1.1

```
893A    11 0082                ld de,130
893D    CD 001B                call drawto
8940    21 0001                ld hl,1
8943    11 00B4                ld de,180
8946    CD 0018                call moveto
8949    21 01FE                ld hl,510
894C    11 00B4                ld de,180
894F    CD 001B                call drawto
8952    CD 881A                call setpen
8955    C9                     ret

8956               clr2frame:              ; da nur zwei linien ist
                                   ; loeschen einfach
8956    CD 8822                call erapen
8959    21 0001                ld hl,1
895C    11 003C                ld de,60
895F    CD 0018                call moveto
8962    21 01FE                ld hl,510
8965    11 003C                ld de,60
8968    CD 001B                call drawto
896B    21 0001                ld hl,1
896E    11 006E                ld de,110
8971    CD 0018                call moveto
8974    21 01FE                ld hl,510
8977    11 006E                ld de,110
897A    CD 001B                call drawto
897D    CD 881A                call setpen
8980    C9                     ret


8981               count:          ; ix -> bcdspeicher , um b erhoehen
8981    DD 7E 00               ld a,(ix+0)      ; lsb
8984    80                     add a,b
8985    27                     daa
8986    DD 77 00               ld (ix+0),a
8989    D0                     ret nc
898A    DD 7E 01               ld a,(ix+1)
898D    C6 01                  add a,1
898F    27                     daa     ; uebertrag
8990    DD 77 01               ld (ix+1),a
8993    C9                     ret

8994               cnt525:         ; 5 1/4 Zaehler mit merker
8994    C5                     push bc          ; Register nicht zerstoeren
8995    06 05                  ld b,5           ; ix-> Bcd-Zaehler
8997    CD 8981                call count
899A    3A 8401                ld a,(merker)
899D    C6 01                  add a,1          ; 0,1,2,3 erlaubt
899F    32 8401                ld (merker),a
```

zu Abb. 9.1.1

```
9A2   FE 04                  cp 4
                     ;1      if z
9A4   C2 89B0                JP NZ,.L7
9A7   AF                     xor a
9A8   32 8401                ld (merker),a
9AB   06 01                  ld b,1
9AD   CD 8981                call count
                     ;1      endif
9B0                  .L7:
9B0   C1                     pop bc
9B1   C9                     ret


9B2                  messper:              ; Messen ersten Wechsel  ----___----
                                   ; nur Kanal 1
                                   ; eine Periode in ys. Dezimal
                                   ; Kanal 1 und 2  (bit 0,1)
9B2   AF                     xor a
9B3   32 8401                ld (merker),a   ; 1/4 Zaehler ruecksetzen
9B6   21 000B                ld hl,11        ; ca.11ys sind am Anfang mit Trigger vergange
9B9   22 8402                ld (bcd0),hl
9BC   21 8456                ld hl,data      ;
9BF   DD 21 8402             ld ix,bcd0      ; Erste Zahl
9C3   01 0200                ld bc,512       ; MAX Suchvorgang
                     ;1      loop   ;------_______  Messen
9C6   E5                     PUSH HL
9C7   21 8A07                LD HL,.L8
9CA   E3                     EX (SP),HL
9CB                  .L9:
9CB   7E                     ld a,(hl)       ; Datenwert Bit 0
9CC   E6 01                  and 1   ; pruefen
                     ;2      if z
9CE   C2 89F1                JP NZ,.L10
                     ;3      loop
9D1   E5                     PUSH HL
9D2   21 89F0                LD HL,.L11
9D5   E3                     EX (SP),HL
9D6                  .L12:
9D6   7E                     ld a,(hl)
9D7   E6 01                  and 1
                     ;4      exitif nz       ; Ende Periodendauer
9D9   C0                     RET NZ
9DA   CD 8994                call cnt525
9DD   23                     inc hl
9DE   0B                     dec bc
                     ;4      if bc=0
9DF   79                     LD A,C
9E0   B0                     OR B
9E1   C2 89ED                JP NZ,.L13
9E4   3E FF                  ld a,0ffh
9E6   DD 77 00               ld (ix+0),a
9E9   DD 77 01               ld (ix+1),a
```

zu Abb. 9.1.1

```
                        ;5      exit
89EC    C9                      RET
                        ;4      endif
89ED                    .L13:
                        ;3      endloop
89ED    C3 89D6                 JP .L12
89F0                    .L11:
                        ;3      exit
89F0    C9                      RET
                        ;2      endif
89F1                    .L10:
89F1    CD 8994                 call cnt525
89F4    23                      inc hl
89F5    0B                      dec bc
                        ;2      if bc=0
89F6    79                      LD A,C
89F7    B0                      OR B
89F8    C2 8A04                 JP NZ,.L14
89FB    3E FF                   ld a,0ffh
89FD    DD 77 00                ld (ix+0),a
8A00    DD 77 01                ld (ix+1),a
                        ;3      exit
8A03    C9                      RET
                        ;2      endif
8A04                    .L14:
                        ;1      endloop
8A04    C3 89CB                 JP .L9
8A07                    .L8:
8A07    C9                      ret


8A08                    mess0time:              ; Messen ersten Wechsel ----__----
                                        ; eine Periode in ys. Dezimal
                                        ; Kanal 1 und 2  (bit 0,1)
8A08    AF                      xor a
8A09    32 8401                 ld (merker),a   ; 1/4 Zaehler ruecksetzen
8A0C    21 0000                 ld hl,0         ; hier 0ys delay
8A0F    22 8402                 ld (bcd0),hl
8A12    21 8456                 ld hl,data      ;
8A15    DD 21 8402              ld ix,bcd0      ; Erste Zahl
8A19    01 0200                 ld bc,512       ; MAX Suchvorgang
                        ;1      loop    ;------_______  Messen
8A1C    E5                      PUSH HL
8A1D    21 8A5A                 LD HL,.L15
8A20    E3                      EX (SP),HL
8A21                    .L16:
8A21    7E                      ld a,(hl)       ; Datenwert Bit 0
8A22    E6 01                   and 1   ; pruefen
                        ;2      if z
8A24    C2 8A47                 JP NZ,.L17
                        ;3      loop
```

zu Abb. 9.1.1

```
8A27    E5                      PUSH HL
8A28    21 8A46                 LD HL,.L18
8A2B    E3                      EX (SP),HL
8A2C                    .L19:
8A2C    7E                      ld a,(hl)
8A2D    E6 01                   and 1
                        ;4      exitif nz       ; Ende Periodendauer
8A2F    C0                      RET NZ
8A30    CD 8994                 call cnt525
8A33    23                      inc hl
8A34    0B                      dec bc
                        ;4      if bc=0
8A35    79                      LD A,C
8A36    B0                      OR B
8A37    C2 8A43                 JP NZ,.L20
8A3A    3E FF                   ld a,0ffh       ;
8A3C    DD 77 00                ld (ix+0),a
8A3F    DD 77 01                ld (ix+1),a     ; ???? Ausgabe
                        ;5      exit
8A42    C9                      RET
                        ;4      endif
8A43                    .L20:
                        ;3      endloop
8A43    C3 8A2C                 JP .L19
8A46                    .L18:
                        ;3      exit
8A46    C9                      RET
                        ;2      endif   ; in Hauptschleife nicht zaehlen
8A47                    .L17:
8A47    23                      inc hl
8A48    0B                      dec bc
                        ;2      if bc=0
8A49    79                      LD A,C
8A4A    B0                      OR B
8A4B    C2 8A57                 JP NZ,.L21
8A4E    3E FF                   ld a,0ffh
8A50    DD 77 00                ld (ix+0),a
8A53    DD 77 01                ld (ix+1),a
                        ;3      exit
8A56    C9                      RET
                        ;2      endif
8A57                    .L21:
                        ;1      endloop
8A57    C3 8A21                 JP .L16
8A5A                    .L15:
8A5A    C9                      ret


8A5B                    bcdaus:         ; ix-> bcdstand
                                        ; hl ->Zielbuffer
```

zu Abb. 9.1.1

```
8A5B    DD 7E 01                  ld a, (ix+1)      ; MSB first
8A5E    0F                        rrca
8A5F    0F                        rrca
8A60    0F                        rrca
8A61    0F                        rrca
8A62    E6 0F                     and 0fh
8A64    CD 8A86                   call dezaus
8A67    DD 7E 01                  ld a, (ix+1)
8A6A    E6 0F                     and 0fh
8A6C    CD 8A86                   call dezaus
8A6F    DD 7E 00                  ld a, (ix+0)      ; LSB dann
8A72    0F                        rrca
8A73    0F                        rrca
8A74    0F                        rrca
8A75    0F                        rrca
8A76    E6 0F                     and 0fh
8A78    CD 8A86                   call dezaus
8A7B    DD 7E 00                  ld a, (ix+0)
8A7E    E6 0F                     and 0fh
8A80    CD 8A86                   call dezaus
8A83    36 00                     ld (hl),0         ; 0=Ende String
8A85    C9                        ret

8A86                      dezaus:
8A86    F6 30                     or '0'
8A88    77                        ld (hl),a
8A89    23                        inc hl
8A8A    C9                        ret


8A8B                      ausperiode:
8A8B    CD 89B2                   call messper      ; Messwert nach bcd0
8A8E    DD 21 8ABB                ld ix,txt2        ; Text ausgeben
8A92    CD 8B00                   call txtprint
8A95    DD 21 8402                ld ix,bcd0        ; data
8A99    CD 8A5B                   call bcdaus       ; ausgabe in BCD
8A9C    21 8406                   ld hl,buffer
8A9F    CD 001E                   call textaus      ; und ausgeben auf den Bildschirm
8AA2    C9                        ret


8AA3                      auslow:
8AA3    CD 8A08                   call mess0time
8AA6    DD 21 8ADA                ld ix,txt3
8AAA    CD 8B00                   call txtprint
8AAD    DD 21 8402                ld ix,bcd0
8AB1    CD 8A5B                   call bcdaus       ; 0-Zeitdauer
8AB4    21 8406                   ld hl,buffer
8AB7    CD 001E                   call textaus
8ABA    C9                        ret
```

zu Abb. 9.1.1

```
8ABB                    txt2:           ; fuer Dauer
8ABB    0104 005A               defw 140+10*12,90
8ABF    22 00                   defb 22h,0
8AC1    00                      defb 0


8AC2                    txt1:
8AC2    008C 005A               defw 140,90
8AC6    22 00                   defb 22h,0
8AC8    50 65 72 69             defb 'Periode = ???? ys',0
8ACC    6F 64 65 20
8AD0    3D 20 3F 3F
8AD4    3F 3F 20 79
8AD8    73 00



8ADA                    txt3:
8ADA    0112 00D2               defw 70+17*12,210
8ADE    22 00                   defb 22h,0
8AE0    00                      defb 0

8AE1                    txt4:
8AE1    0046 00D2               defw 70,210
8AE5    22 00                   defb 22h,0
8AE7    30 2D 53 69             defb '0-Signal-Dauer = ???? ys',0
8AEB    67 6E 61 6C
8AEF    2D 44 61 75
8AF3    65 72 20 3D
8AF7    20 3F 3F 3F
8AFB    3F 20 79 73
8AFF    00


8B00                    txtprint:               ; ix->Text
8B00    21 8406                 ld hl,buffer
8B03    06 06                   ld b,6          ; parameter
8B05                    lp1:
8B05    DD 7E 00                ld a,(ix+0)
8B08    77                      ld (hl),a
8B09    DD 23                   inc ix
8B0B    23                      inc hl
8B0C    10 F7                   djnz lp1
                                        ; nun daten
8B0E                    lp:
8B0E    DD 7E 00                ld a,(ix+0)
8B11    77                      ld (hl),a
8B12    B7                      or a
8B13    C8                      ret z   ; 0 = Ende hl bleibt auf Ziel
8B14    23                      inc hl
8B15    DD 23                   inc ix
8B17    18 F5                   jr lp   ; bis Textende
```

zu Abb. 9.1.1

```
8B19                         wartesync:
8B19    CD 882A                      call sync
8B1C    38 FB                        jr c,wartesync
8B1E    C9                           ret

8B1F                         abltxt:         ; Text fuer beide Bildseiten Abgleich 1
8B1F    DD 21 8AC2                   ld ix,txtl      ; Sowie Bildausgaben
8B23    CD 8B00                      call txtprint
8B26    21 8406                      ld hl,buffer
8B29    CD 001E                      call textaus
8B2C    DD 21 8B72                   ld ix,txtal     ; Sowie Bildausgaben
8B30    CD 8B00                      call txtprint
8B33    21 8406                      ld hl,buffer
8B36    CD 001E                      call textaus
8B39    CD 8803                      call wait
8B3C    3E 01                        ld a,00000001b
8B3E    D3 72                        out (gdp+2),a
8B40    21 0000                      ld hl,0
8B43    11 0078                      ld de,120
8B46    CD 0018                      call moveto
8B49    21 01FF                      ld hl,511
8B4C    CD 001B                      call drawto
8B4F    21 01FF                      ld hl,511
8B52    11 00BE                      ld de,190
8B55    CD 001B                      call drawto
8B58    21 0000                      ld hl,0
8B5B    11 00BE                      ld de,190
8B5E    CD 001B                      call drawto
8B61    21 0000                      ld hl,0
8B64    11 0078                      ld de,120
8B67    CD 001B                      call drawto     ; Rahmen
8B6A    CD 8803                      call wait
8B6D    3E 00                        ld a,0
8B6F    D3 72                        out (gdp+2),a
8B71    C9                           ret

8B72                         txtal:
8B72    0000 0000                    defw 0,0
8B76    22 00                        defb 22h,0
8B78    53 3D 53 70                  defb 'S=Speichern W=Weiter M=Menue',0
8B7C    65 69 63 68
8B80    65 72 6E 20
8B84    57 3D 57 65
8B88    69 74 65 72
8B8C    20 4D 3D 4D
8B90    65 6E 75 65
8B94    00

8B95                         abgleich1:                 ; Scop  Kanal 0 darstellen
8B95    CD 0030                      call clrall        ; seiten loeschen zuerst
8B98    3E 50                        ld a,01010000b
8B9A    CD 8812                      call setpage
8B9D    CD 8B1F                      call abltxt
```

zu Abb. 9.1.1

```
8BA0    3E 00                   ld a,00000000b
8BA2    CD 8812                 call setpage    ; Startseite
8BA5    CD 8B1F                 call abltxt
8BA8                    abgl1:
8BA8    CD 892B                 call clrlframe
8BAB    CD 88BB                 call getframe   ; daten laden
8BAE    CD 88D1                 call putlframe  ; ausgabe auf akt schreibseite
8BB1    3E 40                   ld a,01000000b  ; schreib = 1 nun
8BB3    CD 8812                 call setpage
8BB6    CD 8A8B                 call ausperiode         ; beide Seiten
8BB9    CD 892B                 call clrlframe  ; loeschen seite 1
8BBC    CD 88BB                 call getframe
8BBF    CD 88D1                 call putlframe  ; darstellen seite 1
8BC2    3E 10                   ld a,00010000b  ; schreiben seite 0
8BC4    CD 8812                 call setpage
8BC7    CD 8A8B                 call ausperiode         ; Periodendauer ausgeben
8BCA    CD 0027                 call csts
8BCD    28 D9                   jr z,abgl1      ; wieder von vorne
8BCF    CD 0024                 call ci
8BD2    FE 4D                   cp 'M'
8BD4    C8                      ret z   ; Menue
8BD5    FE 6D                   cp 'm'
8BD7    C8                      ret z
                        ;1      if a='S' or a='s'
8BD8    FE 53                   CP 'S'
8BDA    CA 8BE2                 JP Z,.L23
8BDD    FE 73                   CP 's'
8BDF    C2 8BF5                 JP NZ,.L22
8BE2                    .L23:
                        ;2      repeat
8BE2                    .L24:
8BE2    CD 0024                 call ci
                        ;3      exitif a='M' or a='m'
8BE5    FE 4D                   CP 'M'
8BE7    C8                      RET Z
8BE8    FE 6D                   CP 'm'
8BEA    C8                      RET Z
                        ;2      until a='W' or a='w'
8BEB    FE 57                   CP 'W'
8BED    CA 8BF5                 JP Z,.L25
8BF0    FE 77                   CP 'w'
8BF2    C2 8BE2                 JP NZ,.L24
8BF5                    .L25:
                        ;1      endif
8BF5                    .L22:
8BF5    18 B1                   jr abgl1        ; weiter sonst


8BF7                    ab2txt:
8BF7    DD 21 8AE1              ld ix,txt4      ; Sowie Bildausgaben
```

zu Abb. 9.1.1

```
8BFB     CD 8B00                  call txtprint
8BFE     21 8406                  ld hl,buffer
8C01     CD 001E                  call textaus
8C04     DD 21 8B72               ld ix,txtal      ; Sowie Bildausgaben
8C08     CD 8B00                  call txtprint
8C0B     21 8406                  ld hl,buffer
8C0E     CD 001E                  call textaus
8C11     CD 8803                  call wait
8C14     3E 01                    ld a,00000001b
8C16     D3 72                    out (gdp+2),a
8C18     21 0000                  ld hl,0
8C1B     11 0032                  ld de,50
8C1E     CD 0018                  call moveto
8C21     21 01FF                  ld hl,511
8C24     CD 001B                  call drawto
8C27     21 01FF                  ld hl,511
8C2A     11 00BE                  ld de,190
8C2D     CD 001B                  call drawto
8C30     21 0000                  ld hl,0
8C33     11 00BE                  ld de,190
8C36     CD 001B                  call drawto
8C39     21 0000                  ld hl,0
8C3C     11 0032                  ld de,50
8C3F     CD 001B                  call drawto      ; Rahmen
8C42     21 0000                  ld hl,0
8C45     11 0078                  ld de,120
8C48     CD 0018                  call moveto      ; Mittelline
8C4B     21 01FF                  ld hl,511
8C4E     11 0078                  ld de,120
8C51     CD 001B                  call drawto
8C54     CD 8803                  call wait
8C57     3E 00                    ld a,0
8C59     D3 72                    out (gdp+2),a
8C5B     C9                       ret


8C5C                      abgleich2:               ; Scop  Kanal 0 darstellen
8C5C     CD 0030                  call clrall      ; seiten loeschen zuerst
8C5F     3E 50                    ld a,01010000b
8C61     CD 8812                  call setpage
8C64     CD 8BF7                  call ab2txt
8C67     3E 00                    ld a,00000000b
8C69     CD 8812                  call setpage     ; Startseite
8C6C     CD 8BF7                  call ab2txt
8C6F                      abgl2:
8C6F     CD 892B                  call clr1frame
8C72     CD 8956                  call clr2frame
8C75     CD 88BB                  call getframe    ; daten laden
8C78     CD 88D1                  call put1frame   ; ausgabe auf akt schreibseite
8C7B     CD 88FE                  call put2frame
```

zu Abb. 9.1.1

```
8C7E    3E 40                   ld a,01000000b  ; schreib = 1 nun
8C80    CD 8812                 call setpage
8C83    CD 8AA3                 call auslow
8C86    CD 892B                 call clr1frame  ; loeschen seite 1
8C89    CD 8956                 call clr2frame
8C8C    CD 88BB                 call getframe
8C8F    CD 88D1                 call put1frame  ; darstellen seite 1
8C92    CD 88FE                 call put2frame
8C95    3E 10                   ld a,00010000b  ; schreiben seite 0
8C97    CD 8812                 call setpage
8C9A    CD 8AA3                 call auslow     ;Zeitdauer 0-Signal
8C9D    CD 0027                 call csts
8CA0    28 CD                   jr z,abgl2
8CA2    CD 0024                 call ci
8CA5    FE 4D                   cp 'M'
8CA7    C8                      ret z   ; Menue
8CA8    FE 6D                   cp 'm'
8CAA    C8                      ret z
                        ;1      if a='S' or a='s'
8CAB    FE 53                   CP 'S'
8CAD    CA 8CB5                 JP Z,.L27
8CB0    FE 73                   CP 's'
8CB2    C2 8CC8                 JP NZ,.L26
8CB5                    .L27:
                        ;2      repeat
8CB5                    .L28:
8CB5    CD 0024                 call ci
                        ;3      exitif a='M' or a='m'
8CB8    FE 4D                   CP 'M'
8CBA    C8                      RET Z
8CBB    FE 6D                   CP 'm'
8CBD    C8                      RET Z
                        ;2      until a='W' or a='w'
8CBE    FE 57                   CP 'W'
8CC0    CA 8CC8                 JP Z,.L29
8CC3    FE 77                   CP 'w'
8CC5    C2 8CB5                 JP NZ,.L28
8CC8                    .L29:
                        ;1      endif
8CC8                    .L26:
8CC8    18 A5                   jr abgl2


8CCA                    menueein:
                        ;1      repeat
8CCA                    .L30:
                        ;2      repeat
8CCA                    .L31:
8CCA    21 0032                 ld hl,50
8CCD    22 8406                 ld (buffer),hl
8CD0    21 000A                 ld hl,10
8CD3    22 8408                 ld (buffer+2),hl
```

zu Abb. 9.1.1

```
8CD6    3E 33                  ld a,33h
8CD8    32 840A                ld (buffer+4),a
8CDB    3E 00                  ld a,0
8CDD    32 840B                ld (buffer+5),a
8CE0    3E 02                  ld a,2  ; zwei Zeichenfeld
8CE2    32 840C                ld (buffer+6),a
8CE5    3E 00                  ld a,0
8CE7    32 840D                ld (buffer+7),a
8CEA    0E 01                  ld c,1
8CEC    21 8406                ld hl,buffer
8CEF    CD 0021                call textein
8CF2    3A 840D                ld a,(buffer+7)
                       ;2      until a=1
8CF5    FE 01                  CP 1
8CF7    C2 8CCA                JP NZ,.L31
8CFA    3A 840E                ld a,(buffer+8)
                       ;1      until a in ['1'..'4']
8CFD    FE 35                  CP '4'+1
8CFF    D2 8CCA                JP NC,.L30
8D02    FE 31                  CP '1'+0
8D04    DA 8CCA                JP C,.L30
8D07                   .L32:
8D07    C9                     ret

8D08                   help:
8D08    CD 0030                call clrall
8D0B    DD 21 8D6C             ld ix,htxt1
8D0F    11 00EB                ld de,255-10*2
                       ;1      loop
8D12    E5                     PUSH HL
8D13    21 8D5E                LD HL,.L33
8D16    E3                     EX (SP),HL
8D17                   .L34:
8D17    21 0000                ld hl,0          ;x=0, y=variabel
8D1A    22 8406                ld (buffer),hl
8D1D    EB                     ex de,hl
8D1E    22 8408                ld (buffer+2),hl
8D21    EB                     ex de,hl
8D22    3E 22                  ld a,22h
8D24    32 840A                ld (buffer+4),a
8D27    3E 00                  ld a,0
8D29    32 840B                ld (buffer+5),a
8D2C    21 840C                ld hl,buffer+6
8D2F    DD 7E 00               ld a,(ix+0)
                       ;2      while a<>0ah
8D32                   .L35:
8D32    FE 0A                  CP 0ah
8D34    CA 8D43                JP Z,.L36
                       ;3      exitif a=0
8D37    B7                     OR A
8D38    C8                     RET Z
8D39    77                     ld (hl),a        ; Ablegen
```

zu Abb. 9.1.1

```
8D3A    23                      inc hl
8D3B    DD 23                   inc ix
8D3D    DD 7E 00                ld a,(ix+0)
                        ;2      endwhile
8D40    C3 8D32                 JP .L35
8D43                    .L36:
8D43    DD 23                   inc ix
8D45    36 00                   ld (hl),0
8D47    21 8406                 ld hl,buffer
8D4A    D5                      push de
8D4B    DD E5                   push ix
8D4D    CD 001E                 call textaus     ; ausgeben
8D50    DD E1                   pop ix
8D52    D1                      pop de
8D53    21 0014                 ld hl,10*2
8D56    EB                      ex de,hl
8D57    AF                      xor a
8D58    ED 52                   sbc hl,de
8D5A    EB                      ex de,hl
                        ;1      endloop
8D5B    C3 8D17                 JP .L34
8D5E                    .L33:
                        ;1      repeat
8D5E                    .L37:
8D5E    CD 0024                 call ci
                        ;1      until a='M' or a='m'
8D61    FE 4D                   CP 'M'
8D63    CA 8D6B                 JP Z,.L38
8D66    FE 6D                   CP 'm'
8D68    C2 8D5E                 JP NZ,.L37
8D6B                    .L38:
8D6B    C9                      ret



8D6C                    htxtl:
8D6C    20 20 20 20             defb '          *** Rev 1.0 *** ',0ah
8D70    20 20 20 20
8D74    20 20 2A 2A
8D78    2A 20 52 65
8D7C    76 20 31 2E
8D80    30 20 2A 2A
8D84    2A 20 0A
8D87    31 2E 20 49             defb '1. IOE-Karte  auf Adresse 30h legen.',0ah
8D8B    4F 45 2D 4B
8D8F    61 72 74 65
8D93    20 20 61 75
8D97    66 20 41 64
8D9B    72 65 73 73
8D9F    65 20 33 30
8DA3    68 20 6C 65
8DA7    67 65 6E 2E
```

zu Abb. 9.1.1

```
8DAB    0A
8DAC    32 2E 20 49          defb '2. I/O 0-Port Bit 0 = Kanal 1.',0ah
8DB0    2F 4F 20 30
8DB4    2D 50 6F 72
8DB8    74 20 42 69
8DBC    74 20 30 20
8DC0    3D 20 4B 61
8DC4    6E 61 6C 20
8DC8    31 2E 0A
8DCB    33 2E 20 49          defb '3. I/O 0-Port Bit 1 = Kanal 2.',0ah
8DCF    2F 4F 20 30
8DD3    2D 50 6F 72
8DD7    74 20 42 69
8DDB    74 20 31 20
8DDF    3D 20 4B 61
8DE3    6E 61 6C 20
8DE7    32 2E 0A
8DEA    34 2E 20 4B          defb '4. Kanal 1 ist auch Triggereingang.',0ah
8DEE    61 6E 61 6C
8DF2    20 31 20 69
8DF6    73 74 20 61
8DFA    75 63 68 20
8DFE    54 72 69 67
8E02    67 65 72 65
8E06    69 6E 67 61
8E0A    6E 67 2E 0A
8E0E    35 2E 20 44          defb '5. Das Signal erscheint erst',0ah
8E12    61 73 20 53
8E16    69 67 6E 61
8E1A    6C 20 65 72
8E1E    73 63 68 65
8E22    69 6E 74 20
8E26    65 72 73 74
8E2A    0A
8E2B    20 20 20 6E          defb '   nachdem ein Signalwechsel',0ah
8E2F    61 63 68 64
8E33    65 6D 20 65
8E37    69 6E 20 53
8E3B    69 67 6E 61
8E3F    6C 77 65 63
8E43    68 73 65 6C
8E47    0A
8E48    20 20 20 61          defb '   am Triggereingang von 0 auf 1',0ah
8E4C    6D 20 54 72
8E50    69 67 67 65
8E54    72 65 69 6E
8E58    67 61 6E 67
8E5C    20 76 6F 6E
8E60    20 30 20 61
8E64    75 66 20 31
8E68    0A
8E69    20 20 20 65          defb '   erfolgt.',0ah
```

zu Abb. 9.1.1

```
8E6D    72 66 6F 6C
8E71    67 74 2E 0A
8E75    36 2E 20 44              defb '6. Die Messungen erfolgen auf ca. ',0ah
8E79    69 65 20 4D
8E7D    65 73 73 75
8E81    6E 67 65 6E
8E85    20 65 72 66
8E89    6F 6C 67 65
8E8D    6E 20 61 75
8E91    66 20 63 61
8E95    2E 20 0A
8E98    20 20 20 35              defb '   5ys bis 6ys genau.',0ah
8E9C    79 73 20 62
8EA0    69 73 20 36
8EA4    79 73 20 67
8EA8    65 6E 61 75
8EAC    2E 0A
8EAE    20 20 20 20              defb '                      M=Menue',0ah
8EB2    20 20 20 20
8EB6    20 20 20 20
8EBA    20 20 20 20
8EBE    20 20 20 20
8EC2    20 20 4D 3D
8EC6    4D 65 6E 75
8ECA    65 0A
8ECC    00                       defb 0



                                          ; Hauptprogramm


8ECD                     start:
8ECD    31 87FF                  ld sp,stack      ; Stackpointer init
8ED0    AF                       xor a
8ED1    32 8400                  ld (synstate),a
8ED4    CD 0030                  call clrall
8ED7    3E 00                    ld a,0
8ED9    CD 8812                  call setpage
8EDC    21 8F20                  ld hl,meld1
8EDF    CD 001E                  call textaus
8EE2    21 8F37                  ld hl,meld2
8EE5    CD 001E                  call textaus
8EE8    21 8F5C                  ld hl,meld3
8EEB    CD 001E                  call textaus
8EEE    21 8F81                  ld hl,meld4
8EF1    CD 001E                  call textaus
8EF4    CD 8CCA                  call menueein    ; 1..4 in Akku
                         ;1       if a='1'
8EF7    FE 31                    CP '1'
8EF9    C2 8F02                  JP NZ,.L39
```

zu Abb. 9.1.1

```
8EFC    CD 8B95                  call abgleich1
                         ;1      elseif a='2'
8EFF    C3 8F1D                  JP .L40
8F02                     .L39:
8F02    FE 32                    CP '2'
8F04    C2 8F0D                  JP NZ,.L41
8F07    CD 8C5C                  call abgleich2
                         ;1      elseif a='3'
8F0A    C3 8F1D                  JP .L40
8F0D                     .L41:
8F0D    FE 33                    CP '3'
8F0F    C2 8F18                  JP NZ,.L42
8F12    CD 8D08                  call help
                         ;1      elseif a='4'
8F15    C3 8F1D                  JP .L40
8F18                     .L42:
8F18    FE 34                    CP '4'
8F1A    C2 8F1D                  JP NZ,.L43
                                         ; frei fuer Erweiterung
                         ;1      endif
8F1D                     .L43:
8F1D                     .L40:
8F1D    C3 8ECD                  jp start



8F20                     meld1:
8F20    0028 00DC                defw 40,220     ; x,y
8F24    44 00                    defb 44h,0      ;Schrifthoehe
8F26    52 44 4B 2D              defb 'RDK-Digital-Scop'
8F2A    44 69 67 69
8F2E    74 61 6C 2D
8F32    53 63 6F 70
8F36    00                       defb 0

8F37                     meld2:
8F37    0032 00A0                defw 50,160     ; x,y
8F3B    22 00                    defb 22h,0
8F3D    31 20 3D 20              defb '1 = Periodendauer,     1-Kanal'
8F41    50 65 72 69
8F45    6F 64 65 6E
8F49    64 61 75 65
8F4D    72 2C 20 20
8F51    20 20 20 31
8F55    2D 4B 61 6E
8F59    61 6C
8F5B    00                       defb 0

8F5C                     meld3:
8F5C    0032 0082                defw 50,130     ; x,y
8F60    22 00                    defb 22h,0
```

zu Abb. 9.1.1

```
8F62    32 20 3D 20              defb '2 = Vergleichsmessung, 2-Kanal'
8F66    56 65 72 67
8F6A    6C 65 69 63
8F6E    68 73 6D 65
8F72    73 73 75 6E
8F76    67 2C 20 32
8F7A    2D 4B 61 6E
8F7E    61 6C
8F80    00                       defb 0

8F81                     meld4:
8F81    0032 0064                defw 50,100     ; x,y
8F85    22 00                    defb 22h,0
8F87    33 20 3D 20              defb '3 = Kurzerklaerung'
8F8B    4B 75 72 7A
8F8F    65 72 6B 6C
8F93    61 65 72 75
8F97    6E 67
8F99    00                       defb 0



                                 end


Macros:

Symbols:
8854  .L1          89F1  .L10         89F0  .L11
89D6  .L12         89ED  .L13         8A04  .L14
8A5A  .L15         8A21  .L16         8A47  .L17
8A46  .L18         8A2C  .L19         8875  .L2
8A43  .L20         8A57  .L21         8BF5  .L22
8BE2  .L23         8BE2  .L24         8BF5  .L25
8CC8  .L26         8CB5  .L27         8CB5  .L28
8CC8  .L29         88DE  .L3          8CCA  .L30
8CCA  .L31         8D07  .L32         8D5E  .L33
8D17  .L34         8D32  .L35         8D43  .L36
8D5E  .L37         8D6B  .L38         8F02  .L39
88EC  .L4          8F1D  .L40         8F0D  .L41
8F18  .L42         8F1D  .L43         890B  .L5
8919  .L6          89B0  .L7          8A07  .L8
89CB  .L9          8B1F  AB1TXT       8BF7  AB2TXT
8BA8  ABGL1        8C6F  ABGL2        8B95  ABGLEICH1
8C5C  ABGLEICH2    8AA3  AUSLOW       8A8B  AUSPERIODE
8402  BCD0         8404  BCD1         8A5B  BCDAUS
8406  BUFFER       0024  CI           892B  CLR1FRAME
8956  CLR2FRAME    0030  CLRALL       0033  CLRINVIS
880C  CMD          8994  CNT525       8981  COUNT
0027  CSTS         8456  DATA         8A86  DEZAUS
```

zu Abb. 9.1.1

```
001B   DRAWTO        0006   DREHE         0012   ENDSCHLEIFE
8822   ERAPEN        0070   GDP           88BB   GET1
88C0   GET2          88BB   GETFRAME      8842   GITTER
0009   HEBE          8D08   HELP          8D6C   HTXT1
0030   IOE           8B0E   LP            8B05   LP1
8F20   MELD1         8F37   MELD2         8F5C   MELD3
8F81   MELD4         8CCA   MENUEEIN      8401   MERKER
8A08   MESSOTIME     89B2   MESSPER       0018   MOVETO
0060   PAGE          002D   POO           88D1   PUT1FRAME
88FE   PUT2FRAME     002A   RI            000F   SCHLEIFE
0003   SCHREITE      000C   SENKE         8812   SETPAGE
881A   SETPEN        0015   SETZE         87FF   STACK
8ECD   START         883C   SY1           882A   SYNC
8400   SYNSTATE      0000   TEST          001E   TEXTAUS
0021   TEXTEIN       8AC2   TXT1          8ABB   TXT2
8ADA   TXT3          8AE1   TXT4          8B72   TXTA1
8B00   TXTPRINT      8803   WAIT          8804   WAIT11
8B19   WARTESYNC
```

```
No Fatal error(s)
```

Abb. 9.1.1 Das Scop-Programm als Assemblerlisting. Es findet auf 8800h Platz und kann dort auch in ein 2K × 8-EPROM gelegt werden, wenn man es auf der SBC 2-Baugruppe verwenden will

10 Literaturverzeichnis

Bücher

[1] ZILOG Z80-Datenbuch, Vertrieb durch Kontron München.
[2] Z80-Assembler Handbuch. Vertrieb durch Kontron München.
[3] *Klein, Rolf-Dieter.* Mikrocomputersysteme, Franzis-Verlag.
[4] *Klein, Rolf-Dieter.* Mikrocomputer Hard- und Softwarepraxis. Franzis-Verlag, München.
[5] *Klein, Rolf-Dieter.* Basic-Interpreter. Franzis-Verlag.
[6] *Klein, Rolf-Dieter.* Was ist Pascal. Franzis-Verlag.
[7] *Klein, Michael.* Z80-Applikationsbuch. Franzis-Verlag, München.
[8] *Leventhale, Lance A.* Z80 Assembly Language Programming. Osborne/McGRAW. Hill. Berkeley, California.
[9] *Blomeyer-Bartenstein, H.-P.* Microcomputertechnik. Ing. W. Hofacker GmbH Verlag, München.
[10] *Wirth, N.* Systematisches Programmieren. Teubner Studienbücher. Stuttgart.
[11] *Wirth, N.* Algorithmen und Datenstrukturen. Teubner Studienbücher. Stuttgart.
[12] *Feichtinger, Herwig.* Microcomputer von A bis Z. Franzis-Verlag, München.
[13] *Feichtinger, Herwig.* Basic für Microcomputer. Franzis-Verlag, München.
[14] *Klein, Rolf-Dieter.* Die Prozessoren 68000 und 68008. Franzis-Verlag, München.
[15] Plate. Betriebssystem CP/M. Franzis-Verlag, München.
[16] Klein, Rolf-Dieter. Microcomputer selbstgebaut und programmiert. Franzis-Verlag, München.

Zeitschriften

[1] MC. Franzis-Verlag, München.
[2] ELEKTRONIK. Franzis-Verlag, München.
[3] BYTE. Byte Publications Inc. Peterborough.
[4] Dr. Dobbs Journal. Menlo Park.
[5] *Klein, Rolf-Dieter.* Längenbestimmung von Z80-Befehlen. ELEKTRONIK, Heft 23, S85..87.1980.
[6] *Klein, Rolf-Dieter.* Basic für 8080-Systeme, Hobbycomputer Sonderheft 1, Franzis-Verlag, München.
[7] Sonderhefte Schritt für Schritt 1 u. 2. Franzis-Verlag, München.

11 Bezugsquellenverzeichnis

Platinen, EPROMs und Bauteile:

Graf-Elektronik Systeme GmbH
Magnusstraße 13
Postfach 1610
8960 Kempten
Tel. 0831/6211 oder 0831/61930
Mailbox: 0831/69330

Elektronikladen
W.-Mellies-Straße 88
4930 Detmold 18
Tel. 05232/8131

Disketten-Basic (HEBAS):

Dr. Hehl Hans
Lindenstr. 20
8059 Wartenberg

Software (Disketten, Literatur):

Franzis-Verlag
Software-Service
Karlstraße 37
8000 München 37
Tel. 089/5117–331

Roboter Bausätze (passend zur IOE):

Microelectronic Kalms & Mürb GmbH
Fasanenweg 2
7570 Baden-Baden 22
Tel. 07223/57047

Z80-Bausteine

Zilog
Vertrieb z. B. Kontron, Erding b. München

Soundgenerator von General Instrument, München

Thomson-Bausteine
Vertrieb z. B. Metronik, München

TTL-Bausteine
Texas Instruments
Vertrieb über viele Fachgeschäfte

12 Terminologie-Verzeichnis

A

Access
Zugriff; Zugriff zum Beispiel auf einen Speicher.

Ada
Eine Programmiersprache, die im Auftrag des amerikanischen Verteidigungsministerium entwickelt wurde. Ada ist eine sehr umfangreiche Programmiersprache, die auch Multitasking-Konzepte usw. enthält.
Beispiel:

```
loop
  select
    accept  READ( X: out ITEM) do
      X := STORED ;
    end READ ;
  or
    accept  WRITE(X : ITEM) do
      STORED := X ;
    end WRITE ;
  end select ;
end loop ;
```

Adresse
Eine Bezeichnung für einen bestimmten Speicherplatz oder Speicherbereich.

Adreßbus
Ein Bus auf den die Adreßleitungen eines Mikroprozessors geführt werden.

Akkumulator
Ein Register, in dem arithmetische und logische Operationen ausgeführt werden können.

Algol
Algorithmic Language. Es handelt sich um eine Programmiersprache für den technisch wissenschaftlichen Bereich.
Programmbeispiel:

```
'BEGIN'
  'REAL' alpha, betha;
  alpha := 3.1;
  INREAL(1,betha);
  OUTREAL(2,betha*alpha);
'END'
```

ALU
Arithmetic Logic Unit, Rechenwerk. In diesem Teil des Rechners werden die arithmetischen und logischen Verknüpfungen ausgeführt.

APL
A Programming Language. Eine Programmiersprache für den technisch wissenschaftlichen Bereich, die Sprache verwendet dabei spezielle Zeichen. Beispiel:

```
    ∇ R←X     WL    Y
[1]     ⍝ Waagrechte Linie einfuegen
[2]     R←(~R ε X)/R←↑(⌊1↑ρ Y),X
[3]     R←(Y,[⎕ IO]'-')[(⎕ IO+1↑ρ Y) ⌊ R;]
    ∇
```

APU
Arithmetic Processor Unit, siehe auch FPU

ASCII
American Standard Code for Information Interchange. Wird auch mit ISO-7-Bit-Code bezeichnet (DIN 66003). Codierungsart für Zeichen.

Assembler
Ein Übersetzungsprogramm, das aus einem mnemotechnischen Programm-Code einen Maschinen-Code erstellt

B

Bankselekt
Unter einer Speicher-Bank versteht man zum Beispiel eine Gruppe von Speichern. Selekt steht für Auswahl. Bankselekt bedeutet also die Auswahl einer Gruppe von Speichern. Man verwendet ein Bankselekt-Signal zum Beispiel zum Erweitern des Adreßraums

BAS-Mischer
Aus dem Synchron- und Videosignal wird durch elektrisches Mischen ein Signal gewonnen, das beide Signale auf einer Leitung transportieren kann. Dieses BAS-Signal ist zur Ansteuerung von vielen Monitoren geeignet.

Basic
Beginners All Purpose Symbolic Instruction Code. Eine einfache Programmiersprache, die besonders auf Heimcomputern sehr verbreitet ist, jedoch den Nachteil besitzt, nicht strukturiert zu sein.
Beispiel:

```
10 PRINT "Quadratwurzeltabelle"
20 FOR I=1 TO 10
30 PRINT I,SQRT(I)
40 NEXT I
```

Baudrate
Messung des Datenflusses, wobei die Zeit zur Übertragung des kürzesten Elements als Maß genommen wird. Beispiel: 1200 Baud bedeuten eine Übertragung von 1200 Bit pro Sekunde

Baudrate-Generator
Ein Baustein zur Erzeugung eines Taktes, der dann für eine serielle Übertragung verwendet wird.

Betriebssystem
Eine Reihe von Programmen, die es dem Computer ermöglichen, selbstständig Programme zu bearbeiten, CP/M und MSDOS sind z.B. solche Betriebssysteme für Mikrorechner.
Bi-Direktionale Bustreiber
Ein Schaltkreis, der logische Informationen auf einer Leitung in beiden Richtungen übertragen kann.
Bildwiederholspeicher
Die Information, zur Darstellung auf dem Bildschirm wird im Bildwiederholspeicher bereit gehalten, so daß sie fortlaufend ausgegeben werden kann.
Bit
Binary Digit. Kleinste Informationseinheit
Boot
Das Neustarten eines Systems, bei dem Programme geladen werden, wird auch als Boot bezeichnet.
Branch
Verzweigung, Sprung
Buffer
Puffer; Speicher in dem Daten kurzzeitig festgehalten werden, oder auch Treiber zum Schalten von größeren Lasten.
Bus
Sammelleitung, an die mehrere Bausteine angeschlossen werden können. Dabei können auch mehrere Bausteine Daten auf den Bus angegeben, jedoch nicht zur gleichen Zeit. Eine Auswahllogik sorgt dafür, daß immer nur ein Busteilnehmer sendet, wobei jedoch alle weiteren hören dürfen.
Byte
8 Bits werden als 1 Byte zusammengefaßt

C
Eine Programmiersprache, die zum Beispiel mit dem CP/M68k-Betriebssystem geliefert wird. Die Sprache ähnelt sehr der Sprache Pascal. Programmbeispiel:

```
main()
{
  int i;
  float sqrt();
  for (i = 1; i<=10; i++) {
    printf(
      " Wurzel aus %d ist   %f",
      i,sqrt(i));
  }
}
```

Cache
Ein schneller Speicher, der z.B.in dem Prozessor-IC integriert ist, und es z.B. erlaubt kleine Programmschleifen schnell ausführen zu können. Der Prozessor 68020 besitzt z.B. einen Programm-Cache.
Clock
Takt
Cobol
Common Business Oriented Language. Eine Programmiersprache vorwiegend für kaufmännische Probleme. Beispiel:

```
PROCEDURE DIVISION.
START.
 MOVE ZEROS TO N
 MOVE 1 TO FAKULTAET.
 ACCEPT M-EIN FROM
          KARTEN-LESER;
 MOVE M-EIN TO M
...
```

Comal
Common Algorithmic Language; diese Sprache entstand 1973 aus einer Mischung von Basic und Pascal. Sie enthält daher strukturierte Elemente und Parametermechanismen
Programmbeispiel:

```
0010 PROC FENSTER(X,Y) CLOSED
0020    DIM LEERZ$ OF 40
0030    LEERZ$(1:40) := " "
0040    POSI(X,1)
0050    FOR ZN:=1 TO Y-X+1 DO PRINT
          LEERZ$
0060   POSI(X,1)
0070 ENDPROC FENSTER
```

Compiler
Ein Übersetzungsprogramm, das eine höhere Programmiersprache in den Maschinen-Code übersetzt. Siehe Kapitel Pascal/S und Gosi
Conditional
Bedingt
Controller
Steuereinheit
CP/M
Disk Operating System für 8080, 8085, Z80, 8086 und 68000 von DIGITAL RESEARCH. Siehe Kapitel Betriebssysteme
Cross-Assembler
Ein Assembler, der nicht auf der Maschine läuft, für die er Code erzeugt. Zum Beispiel ist ein Assembler für den 68000 ein Cross-Assembler, wenn man ihn unter CP/M80, also z.B. auf dem Z80 laufen lassen kann.
Cross-Compiler
Ein Übersetzer für eine höhere Programmiersprache, der auf einem anderen Prozessor läuft, als für welchen er Maschinen-Code erzeugt.
CRT
Cathode Ray Tube; Datensichtgerät oder Bildschirm

Cursor
Sichtmarke zur Kennzeichnung der aktuellen Schreibposition auf dem Bildschirm.

D

Datenbus
Ein Bus, auf den die Datenleitungen eines Mikroprozessors geführt werden.
Debugging
Wörtlich "entwanzen". Gemeint ist die Fehlersuche und Beseitigung in Programmen.
Decrement
Erniedrigen, herunterzählen
Digit
Ziffer, Stelle
DIL
Dual In Line - Gehäuseform
Direktory
Inhaltsverzeichnis: z.B. von einer Diskette
DMA
Direct Memory Access; direkter Zugriff auf den Speicher eines Rechners, wobei die Zugriffssteuerung von einer Peripherieeinheit vorgenommen wird.
DOS
Disc Operating System, Betriebssystem; siehe Kapitel Betriebssystem
dynamische Speicher
Ein Speicher, bei dem die Speicherzellen ständig angesprochen werden müssen, damit sie ihre Information nicht verlieren.

E

Editor
Ein Programm, das die Eingabe von Text erlaubt.
EEPROM
Electrical Erasable Programmable Read Only Memory. Ein Speicher, der sich elektrisch programmieren und löschen läßt. Dabei bleibt die Information nach dem Ausschalten der Versorgungsspannung erhalten. Im Gegensatz zu den EPROMs werden die EEPROMs durch Anlegen einer höheren Spannung gelöscht.
Emulation
Softwaremäßige Nachbildung eines Computers, so daß der Befehlssatz des einen Computers auf einem anderen verfügbar wird. Für den 68000/8 gibt es einen Z80-Emulator, so daß man Z80 Programme ablaufen lassen kann, obwohl man einen 68000/8 verwendet.
Enable
Freigabe
enter
Eingeben
EPROM
Erasable Programmable Read Only Memory. Ein löschbarer Nur-Lese-Speicher. Siehe Kapitel PROMMER
erase
Löschen
Error
Fehler, Irrtum
Even
bedeutet gerade im Gegensatz zu ungerade
Expression
Ausdruck

Fan-in
Eingangslastfaktor
Fan-out
Ausgangslastfaktor; er gibt an, wieviele Bausteine der gleichen Logikserie an diesem Ausgang angeschlossen werden dürfen.
Festwertspeicher
Ein Speicher, dessen Inhalt nicht (oder nur mit Mühe) geändert werden kann.
Fifo
First In First Out. Zuerst eingehende Daten werden auch zuerst wieder ausgegeben.
File
Datei, Daten. Eine Ansammlung von Datengruppen, die in einer Datei angelegt werden.
Firmware
Eine Software, die fest zur Funktionsfähigkeit eines Systems nötig ist und z.B. in einem ROM abgelegt ist.
Fixed-Point
Festkomma
Flag
Eine Marke oder ein Flip-Flop zum Festhalten eines Zustands.
Floating-Point
Gleitkomma
Forth
Eine Programmiersprache, die auf der UPN (umgekehrt polnische Notation) basiert.
Beispiel:

```
: TEXTAUS ."Hallo Forth Erg="
   + * . ;
3 4 5 TEXTAUS
```

Fortran
Formula Translation. Eine problemorientierte Programmiersprache für den technisch wissenschaftlichen Bereich. Beispiel:

```
      SUBPROGRAM BEISPIEL
       COMPLEX Z1,Z2
C     komplexe Zahlen sind möglich
      READ* A,B,C
      D = (B*B-4.*A*C)/(4.*A*A)
      IF (D.GE.0.) GOTO 20
      Z1 = COMPLX(-B/2.*A),SQRT(-D)
      Z2 = CONJG(Z1)
      PRINT*, 'Z1=',Z1,' Z2=',Z2
20    STOP
      END
```

FPU
Floating Point Unit. Gleitkommarechner. Für den 68020 und 68000/8 gibt es z.B. den Baustein 68881, der eine FPU darstellt.
FSK
Frequency shift. Verfahren bei der Aufzeichnung auf Datenträger.

G

Gate
Verknüpfungsschaltung
GND
Masseanschluß, 0V

H

Handshake
Quittungsbetrieb. Durch Steuersignale werden Geräte mit verschiedenen Arbeitsgeschwindigkeiten synchronisiert.
Hardcopy
Kopie. Zum Beispiel Ausdruck eines Bildschirminhalts
Hardware
Damit sind alle Bauteile, Geräte eines Systems gemeint,
Hexadezimal
Siehe Sedezimal
High order
Höherwertige Stelle

I

ICE
In-Circuit Emulator. Gerät zum Test und Entwicklung von Mikrorechnerschaltungen
Increment
Erhöhen, raufzählen
Initialisierung
Die Anfangsschritte in einem Programm, um definierte Startwerte zu erhalten
Input
Eingabe
Instruktionszyklus
Ablauf eines Befehlsausführungsvorgangs
INT
Interrupt. Unterbrechungsanforderung
Interpreter
z.B. ein Programm, das Befehle einer höheren Programmiersprache direkt ausführt und sie nicht vorher in Maschinensprache übersetzt.
Interrupt
Unterbrechung

J

Job
Auftrag
Joystick
Entweder ein Steuerknüppel mit vier Kontakten in den Endstellungen, oder ein Steuerknüppel mit zwei Potentiometern
Jump
Sprung

K

Keyboard
Tastatur
Kit
Bausatz
kompatibel
Austauschbar, aneinander angepaßt;

L

Label
Marke. In Programmiersprachen ist damit meist eine symbolische Adresse gemeint
Lichtgriffel
Ein Stift mit einem optischen Aufnehmer, der auf den Bildschirm gehalten wird. Der Rechner kann dann die Position des Lichtgriffels ermitteln.
Lifo
Last In First Out. Zuletzt gespeicherte Daten werden zuerst ausgegeben (Stack).
Linker
Ein Programm, das mehrere Teilprogramme, die schon übersetzt wurden zu einem gesamten Programm zusammenfügen kann. Dabei können die Teilprogramme Bezüge untereinander enthalten
Lisp
List Processing. Eine Programmiersprache für die Verarbeitung von Listenstrukturen und rekursiver Technik für Probleme der künstlichen Intelligenz. Beispiel:

```
(DEFLIST
 ((CAAR(LAMBDA(X)(CAR(CAR X))))
  (CADR(LAMBDA(X)(CAR(CDR X))))
  (CDAR(LAMBDA(X)(CDR(CAR X))))
  (CDDR(LAMBDA(X)(CDR(CDR X)))) )
 EXPR )
```

Listing
Ausdruck, Auflistung

Loader
Ein Ladeprogramm
Logik Analysator
Ein Geräte, mit dem man digitale Schaltungen auf ihr Zeitverhalten untersuchen kann. Ferner gibt es für Mikroprozessoren auch die Möglichkeit den Befehlsablauf sichtbar zu machen.
Logo
Eine Programmiersprache, die ännlich wie Lisp auch mit Listen arbeitet, jedoch zusätzlich graphische Ausgabebefehle besitzt. Die Sprache wurde zum Programmieren und Lernen für Kinder entwickelt, bietet jedoch Fähigkeiten, die bis in den Hochschulbereich reichen. Die Sprache gibt es für unterschiedliche Nationalitäten. Eine Besonderheit ist, daß man sich selbst Befehle definieren kann, die dann die Sprache erweitern.
Beispiel:

```
LERNE #UBERSETZE :L
 WENN :L = [ ] DANN RÜCKKEHR
  DEF ERSTES :L #UEB PRLISTE
                          ERSTES :L
 #ÜBERSETZE OHNEERSTES :L
ENDE
LERNE KREIS :N
 WH 360 [ VW :N RE 1]
ENDE
```

Loop
Schleife; durch einen Sprung zurück kann eine Programmschleife entstehen.
Low order
Niederwertige Stelle

M

Mark
Bei der seriellen Übertragungs ist damit der logische Wert 1 gemeint, im Gegensatz zu Space.
Maschinenbefehl
Ein Befehl, den der Computer unmittelbar verstehen kann.
maskieren
Damit kann die Ausführung von Interrupts z.B. gestoppt oder freigegeben werden. Ferner bedeutet maskieren auch bestimmte Bits ausblenden.
Memory
Speicher
Mikrocomputer
Besteht aus einem Mikroprozessor, Speichern und Peripherie.
Mikroprogrammierbar
Der Befehlssatz eines Prozessors kann mit Hilfe von Mikrobefehlen definiert werden. Nicht mit Maschinensprache zu verwechseln. Der 68000/8 enthält z.B. ein Mikroprogramm das in seinem Inneren abgelegt ist und den Befehlssatz definiert, es kann jedoch nicht verändert werden.
Mikroprozessor
Ein integrierter Baustein, als Teil eines Mikrocomputers, der ein Leit- und Rechenwerk besitzt.
Mikrorechner
Ein Computer, der mit einem Mikroprozessor aufgebaut ist.
mnemotechnische Darstellung
Leicht zu merkende Abkürzungen für längere Begriffe, z.B. ist BRA die mnemotechnische Abkürzung für Branch.
Modem
Modulator und Demodulator. Eine Schaltung, die Daten für eine Fernübertragung aufbereitet. Ein Akkustikkoppler ist zum Beispiel ein solches Modem.
Modula 2
Eine Programmiersprache, die alle Konzepte von Pascal enthält, zusätzlich jedoch das Modul-Konzept beinhaltet.
Beispiel:

```
DEFINITION MODULE Buffer;
  EXPORT QUALIFIED ablegen, holen,
    nichtleer, nichtvoll;
  VAR nichtleer, nichtvoll : BOOLEAN;
  PROCEDURE ablegen(x : CARDINAL);
  PROCEDURE abholen(VAR x :
                          CARDINAL);
 END Buffer.
```

Monoflop
Ein bistabiles Speicherelement, das nach dem Auslösen nur eine bestimmte Zeit in dem neuen Zustand bleibt und danach wieder in den Ruhezustand zurückfällt.
Multiplex
Übertragung von mehreren verschiedenen Informationen, die dazu zeitlich hintereinander übertragen werden.
Multiprozessing
Ein aus mehreren CPUs oder Teil-Computern zusammengesetzter Rechner.

N

Nesting
Verschachtelung; z.B. verschachteln von Unterprogrammen.
NMI
Non Maskable Interrupt. Eine Unterbrechung, die nicht gesperrt werden kann.

O

Odd
bedeutet ungerade. Die Zahl 3 ist zum Beispiel ungerade.

offener Kollektor
Schaltung, dessen Endtransistor einen herausgeführten, unbeschalteten Kollektor hat.
Oktal
Zahlendarstellung zur Basis 8
Output
Ausgabe

P

Packen
Dabei werden z.B. zwei Dezimalzahlen in einem Byte untergebracht
Parity
Parität, Gleichheit
Pascal
Eine höhere Programmiersprache, die für Lehrzwecke entworfen wurde und zunehmend Verbreitung findet. Siehe Kapitel Pascal/S
Pass
Lauf oder auch Durchgang, z.B. bei einem Übersetzungsvorgang
PE-Verfahren
Phase encoding. Verfahren bei der Aufzeichnung auf Datenträger, siehe Kapitel CAS
Pegel
Spannungsbereich
Peripherie-Geräte
Einheiten, die mit der Außenwelt eines Computer in Verbindung treten können.
Pipelining
Fließbandverarbeitung. An mehreren Stellen wird in kleinen Schritten gleichzeitig eine Verarbeitung vorgenommen. Dadurch kann man in Mikroprozessoren die Befehlszykluszeiten verkleinern.
PL/1
Programming Language 1. Eine höhere Programmiersprache, die zur Pascal-Familie gehört. Beispiel:

```
TEST: PROCEDURE OPTIONS(MAIN);
 DECLARE (A,B) FIXED DECIMAL
  (6,2),
  (COUNT) FIXED;
 A = 12.34; B = A + 1.02;
 PUT SKIP(2);
 DO COUNT=1 TO 10;
   PUT EDIT('I=',I,'B=',B) (F(5),F(6,2));
   B = B + 1.0;
 END;
END TEST;
```

PL/M
Ähnlich der Programmiersprache PL/1, jedoch eine Teilmenge daraus. Wurde vorwiegend für die 8080-Prozessorfamilie verwendet.
Plotter
Ein Gerät, ähnlich zu einem XY-Schreiber, für die Ausgabe von graphischen Darstellungen.
Pointer
Zeiger. Ein Speicherplatz, der eine Adresse eines anderen Speicherplatzes enthält.
Polling
Aufrufbetrieb. Darunter versteht man die ständige Abfrage, z.B. eines Peripheriegerätes, um so festzustellen, ob es schon fertig ist.
Port
Tor. Man meint damit Ein- oder Ausgabebausteine
Prellen
Mechanische Schalter berühren die Kontaktflächen beim Schließen oder Öffnen mehrere Male.
Programm
Ist eine Folge von Anweisungen (Befehlen), die zur Lösung eines Problems dienen sollen.
Programmiersprache
Eine Sprache zur Formulierung von Programmen, die automatisch (von einem Übersetzungsprogramm oder Interpreter) in Maschinensprache umgesetzt werden können.
Programmspeicher
Dort ist das auszuführende Programm abgelegt.
Programmzähler
Er legt die Adresse der Speicherzelle des nächsten Befehls fest.
PROM
Programmable Read Only Memory. Ein Festwertspeicher, der durch Anlegen elektrischer Impulse beschrieben werden kann.
Pseudobefehl
Eine Instruktion, die nicht im Befehlssatz des Prozessors vorhanden ist, und zur Steuerung des Assemblers dient.
Pull-Up-Widerstand
Ein Widerstand, nach +5V geschaltet, um z.B. eine offene Kollektorschaltung zu beschalten.

Q

Queue
Warteschlange. Daten werden in einer Warteschlage angesammelt, wenn sie noch nicht verarbeitet sind.

R

RAM
Random Access Memory. Speicher mit wahlfreiem Zugriff
Real time clock
Echtzeituhr
Redundanz
Teil einer Nachricht, die zur eigentlichen Information nichts mehr beiträgt.
refresh
Wiederauffrischen

Relokalisierbar
Ein Programm, das in verschiedenen Speicherbereichen lauffähig ist.
Reset
Rücksetzen
ROM
Read Only Memory; ein Speicher, den man nur auslesen kann.

S

scan
Abtasten
scrollen
Der Bildschirminhalt wird nach oben oder unten verschoben.
sedezimal
Zahlendarstellung zur Basis 16
select
Auswählen
sense
Abtasten
Simulator
Ein Programm, das einen Vorgang künstlich nachbildet.
Software
Hierunter versteht man alle Arten von Programmen, wie auch Texte und Informationen.
Source
Quelle
Space
Bei der seriellen Übertragung ist damit der logische Wert 0 gemeint.
Space-Taste
Leertaste auf der Tastatur, die einen Freiraum erzeugt.
Speicherbaustein
Ein Baustein, der Informationen behalten kann.
Stack
Stapelspeicher, Kellerspeicher. Siehe LIFO
State
Zustand
Statement
Anweisung, Befehl
statische RAMs
Speicher, die z.B. mit zwei Transistorzellen aufgebaut sind und wie ein bistabiles Flip-Flop arbeiten.
Steuerwerk
Dieser Teil des Computers kontrolliert die Ausführung sämtlicher Befehle, er wird auch mit Leitwerk bezeichnet.
Strukturierte Programmierung
Verfahrensweise, um einfach zu testende und verständliche Programme zu erzeugen.
Subroutine
Unterprogramm
Supervisor
Ein Organisationsprogramm

T

Tantal-Kondensator
Wird in Mikroprozessorschaltungen gerne zur Unterdrückung von Spannungsspitzen auf der Versorgungsleitung verwendet.
Terminal
Datenendstation
Text-Editor
Siehe Editor
Time sharing
Zeitscheibenverfahren. Dabei können mehrere Benutzer auf ein und denselben Computer zugreifen.
Timing-Diagramm
Zeitlicher Ablauf, bildlich dargestellt.
Trace
Ablaufverfolgung; Methode zur Fehlersuche in Programmen.
transfer
Übertragen
Tristate-Treiber
Ein Schaltkreis, der drei Zustände besitzt. Pegel auf 0, Pegel auf 1 oder offen (Pegel undefiniert).

U

UART
Universal Asynchronous Receiver/ Transmitter.
Die Schaltung dient der seriellen Übertragung.
Unit
Einheit, Gerät
Unterprogramm
Gleiche Befehlsfolgen, die in einem Programm mehrfach vorkommen, kann man zu Unterprogrammen zusammenfassen, und muß sie daher nur einmal abspeichern.

V

V24
Schnittstellen-Norm für serielle Signale.
Valid
gültig
Vektor Interrupt
Das auslösende Gerät gibt zusätzlich zur Interrupt-Anforderung auch noch eine Zieladresse vor.

W

Wait
Warten

Wired-Or
Eine logische Verknüpfung, die nur durch die Verdrahtung entsteht.

Worst case
Ungünstigster Fall

Wort
Zusammenfassung mehrerer Bits zu einer logischen Einheit.

Z

Z80-CPU
Ein Mikroprozessor-Baustein

Zeichengenerator
Der Zeichensatz für die Schriftdarstellung auf dem Bildschirm ist darin gespeichert.

Zugriff
Zugang z.B. in eine bestimmte Speicherzelle

Zyklus
Eine Anzahl von Schritten, die wiederholt werden und im Ablauf gewisse Ähnlichkeiten aufweisen

Sachverzeichnis

Klein

Rechner modular

Wer seinen Computer von Grund auf selbst baut und programmiert, der wird ihn vollständig verstehen. Nach diesem bewährten Konzept geht Rolf-Dieter Klein im vorliegenden Buch und in der gleichnamigen Fernsehsendung vor.

Der Aufbau gelingt dem Leser dank kleiner, überschaubarer Module mit erstaunlicher Sicherheit. Danach führt sein Weg von einfachen Übungen mit dem Grundprogramm über interessante Grafik bis hin zum Einsatz des professionellen Betriebssystems CP/M.

Zum Schluß verfügt der Leser über ein modernes und leistungsfähiges Computersystem. Dieses garantiert durch seinen modularen Aufbau nicht zu veralten und kann leicht erweitert werden.



ISB N 3-7723-8721-7 DM +068.00

